Empirical Software Engineering manuscript No.
(will be inserted by the editor)

Impact of Extensions on Browser Performance: An Empirical Study on Google Chrome

Bihui Jin · Heng Li · Ying Zou

Received: date / Accepted: date

Abstract Web browsers have been used widely by users to conduct various online activities, such as information seeking or online shopping. To improve user experience and extend the functionality of browsers, practitioners provide mechanisms to allow users to install third-party-provided plugins (i.e., extensions) on their browsers. However, little is known about the performance implications caused by such extensions. In this paper, we conduct an empirical study to understand the impact of extensions on the user-perceived performance (i.e., energy consumption and page load time) of Google Chrome, the most popular browser. We study a total of 72 extensions from 11 categories (e.g., Developer Tools and Sports), consisting of 61 extensions with distinct types of privacy practices used and 11 extensions without adopting any privacy practices (i.e., no privacy-related data is collected). We observe that browser performance can be negatively impacted by the use of extensions, even when the extensions are used in unintended circumstances (e.g., when logging into an extension is not granted but required, or when an extension is not used for designated websites). We also identify a set of factors that significantly influence the performance impact of extensions, such as code complexity and privacy practices (i.e., collection of user data) adopted by the extensions. Based on our empirical observations, we provide recommendations for developers and users to mitigate the performance impact of browser extensions, such as conducting performance testing and optimization for unintended usage scenarios of extensions, or adhering to proper usage practices of extensions (e.g., logging into an extension when required).

Bihui Jin
David R. Cheriton School of Computer Science
University of Waterloo, Waterloo, ON, Canada
E-mail: bihui.jin@uwaterloo.ca

Ying Zou
Department of Electrical and Computer Engineering
Queen's University, Kingston, ON, Canada
E-mail: ying.zou@queensu.ca

Heng Li
Department of Computer Engineering and Software Engineering
Polytechnique Montréal, Montréal, QC, Canada
E-mail: heng.li@polymtl.ca

Keywords Browser Performance · Energy Consumption · Response Time · Privacy · Google Chrome · Google Extensions · Performance Measurement and Analysis · Software Engineering

1 Introduction

Online services, such as information seeking, video streaming, or social networking services, rely on web browsers as the user interfaces to allow the users to interact with the provided services. In particular, Internet video dominates global Internet traffic, accounting for 71%, and the web/data category (e.g., media and entertainment services and banking applications) constitutes 12% of all Internet traffic in 2022 (Manner, 2022). With the growth of Information and Communication Technology (ICT), the median size of desktop web pages has surged by 336% in a decade, escalating from 468 KB in 2010 to 2042 KB in 2020 (Manner, 2022). When a web browser takes longer to load a webpage or runs slower, it can be frustrating for users and leads to diminished productivity or even customer attrition (Tian and Ma, 2019; Borgolte and Feamster, 2020; Pourghassemi, Amiri Sani, and Chandramowlishwaran, 2019).

Energy usage is considered as a non-trivial quality attribute of software products (Pang et al., 2016). Over the last three decades, the total energy consumption of ICT has surged by a staggering 822.79%, increasing from 2182.72 TWh per year in 2001 to an estimated amount of 17959.11 TWh per year in 2030 (P. Wang et al., 2022). Energy consumption is a concern for browser users, particularly for users of battery-powered devices (e.g., laptops) (Banerjee et al., 2007; Kor et al., 2015). Web browsers operating on portable devices, such as laptops and mobile phones, drain a substantial amount of power to keep the webpage alive, update the content of webpages, or retain multiple tabs (Macedo et al., 2021). Moreover, the excessive energy consumption of electricity-power devices (e.g., desktops) has significant environmental impact (Amsel and Tomlinson, 2010; Murugesan, 2008). It is preferable that the software can be energy efficient, extend battery life, and enhance the overall user experience (Pang et al., 2016). The current studies (Tiwari et al., 1996; Amsel and Tomlinson, 2010; Fei et al., 2004) on energy consumption tend to predominately focus on CPU usage while neglecting the actual patterns of energy consumption resulting from software.

The performance of web browsers (e.g., energy consumption or page load time) plays a crucial role in shaping user experience and ensuring sustainability. Web browsers, such as Google Chrome, typically support a variety of extensions that allow users to extend the browsers' functionalities, such as advertisement (ad) blocking, password management, and language translation. However, the extensions can consume additional resources, such as processing power, memory, and network bandwidth, and may potentially affect the performance of browsers. Prior work (Pearce, 2020; Merzdovnik et al., 2017; Borgolte and Feamster, 2020) has studied extensively on browser performance. Nonetheless, the prior work has primarily examined the impact of browser extensions on either page load time or energy consumption of particular types of extensions. For example, Pearce (2020) studies the energy effect of three ad blocking extensions in the Chrome browser during page loading and finds that open-source ad blockers reduce the waiting time for ads to load and decrease power consumption. Borgolte and Feamster (2020) study the impact of eight privacy-focused extensions on browser performance in terms of page load time and CPU time. Thus, the results of aforementioned studies may not be generalizable to other categories of extensions.

To address the limitation of the existing work, we offer a comprehensive approach to understand the impact of the extensions on the user-perceived performance of the browser. More specifically, our approach consists of the following aspects:

- We encompass a wider variety of extension types by selecting 72 representative extensions from 11 categories (e.g., shopping, blogging, or accessibility).
- We study performance metrics, including both page load time and energy consumption, as energy consumption and page load time directly impact user experience (Palomba et al., 2018; Chan-Jong-Chu et al., 2020; Janssen et al., 2022; Tian and Ma, 2019; Borgolte and Feamster, 2020; Hindle, 2013).
- We investigate energy consumption by breaking down energy consumption from two phases, namely, page load and stabilized energy consumption, for a detailed analysis.
- We systematically examine how the configurations (e.g., active or inactive) and influential factors (e.g., code metrics) of extensions impact browser performance.

We organize our paper along three research questions (RQs) in a progressive manner. Specifically, RQ1 investigates the overall performance impact of browser extensions. RQ2 further studies the difference brought by various usage modes of the extensions, and RQ3 systematically examines a wide range of factors of extensions to identify the factors that can significantly impact browser performance.

RQ1: How do extensions impact browser performance? Practitioners may not be aware that extensions can impair the performance of browsers and degrade the user experience. In this RQ,we focus on the fully-loaded mode of extensions, in which the extension is configured to perform as intended by the developers[1]. We analyze the impact of extensions on the browser performance when they are used in the expected mode (i.e., fully-loaded mode) and observe that the use of extensions can lead to a statistically significant impact on the browser performance, with the largest negative impact on the load time energy consumption.

RQ2: How do the usage modes of extensions affect browser performance? Users do not always interact with extensions in the intended or desired manner, leading to unexpected usage modes. For example, when using extensions, a user has the option to use the extension after login or use the extension without login. Logged or not logged into the extension are two usage modes. Extensions may have different performance impacts depending on different usage modes. By studying how different usage modes of extensions impact the browser performance, we find that browser performance can be negatively impacted by extensions even when they are used in unexpected circumstances (e.g., not logged into the extension) or are not active (e.g., not used for designated websites). Such unintended usage scenarios may even lead to a worse performance impact than the fully-loaded mode, suggesting the need for performance testing and optimization for such scenarios.

RQ3: What factors of extensions influence browser performance? When developers create extensions, they need to consider various factors that may potentially affect browser performance. In this RQ, we conduct quantitative evaluations on the factors of extensions that could impact browser performance. For instance, we observe that the adopted privacy practices of extensions (e.g., *personal communication* and *website contents*) can significantly impact browser performance, suggesting the need to consider the performance impact of privacy practices when developing and using extensions. Elevated code complexity (e.g., the number of functions) can also contribute to heightened energy consumption. Furthermore, extension developers ought to be mindful that certain file types (e.g., SVG image files and OGG audio files) used in extensions adversely affect browser performance.

In summary, the main contributions of this paper are three-fold:

[1] The fully-loaded mode represents that the extension is configured to enable all the necessary capabilities for the intended usage: i.e., the user is logged in (if required), the extension is used on its designated websites, and it has the necessary permissions granted to access the website.

– We provide empirical evidence of the performance impact of browser extensions and their usage modes.
– We identify the factors that contribute to the performance impact of browser extensions.
– We provide recommendations for developers to conduct performing performance testing and to optimize unintended usage scenarios of extensions, as well as suggest users to adhere to proper usage practices in order to mitigate the performance impact of extensions.

Paper organization. We present the design of experiments in Section 2. We discuss our approaches and results in Section 3. Section 4 discusses the implications of this study. In Section 5, we describe the threats to validity. We give an overview of related works in Section 6. Finally, we conclude this paper and discuss future work in Section 7.

2 Experiment Design

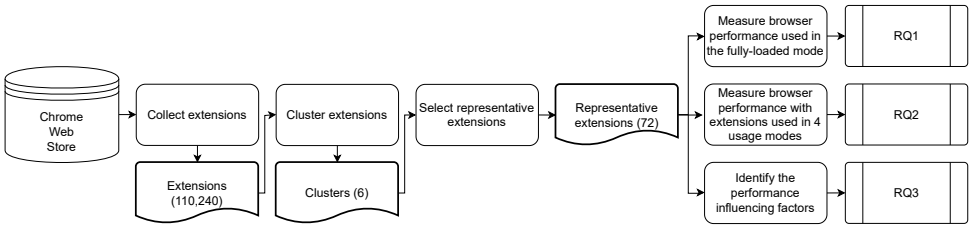

Fig. 1 An overview of our experiment design and analysis.

Figure 1 presents an overview of our approach. We first collect extensions from the Chrome Web Store, then we cluster the extensions according to their similarities and select representative extensions from each cluster as our final studied extensions. With the selected extensions, we design experiments to understand how they impact the browser performance in terms of energy consumption and page load time (RQ1), how different usage modes impact the browser performance (RQ2), and what factors of the extensions impact the browser performance (RQ3).

2.1 Collecting Extensions

Unlike the desktop-based Chrome, the mobile Chrome does not support extensions because mobile devices have limited processing power and storage capacity to support resource-demanding extensions. Therefore, our study targets the desktop-based Chrome, which can be used in portable devices, such as laptops. We collect a total of 110,240 extensions across 11 different categories from the Chrome Web Store. The collected information of the extensions includes the extension name, category, rating score, used privacy practices, extension size, the number of raters, and the number of downloads. An example screenshot of an extension is shown in Figure 2. The descriptive information (e.g., star ratings, category, and number of users) is annotated in the example.

Privacy practices of extensions specify the actions and policies that extensions take to collect the personal information and user data. Extensions can be useful to enhance the functionality of the Chrome browser but can also pose a threat to user privacy and data security.

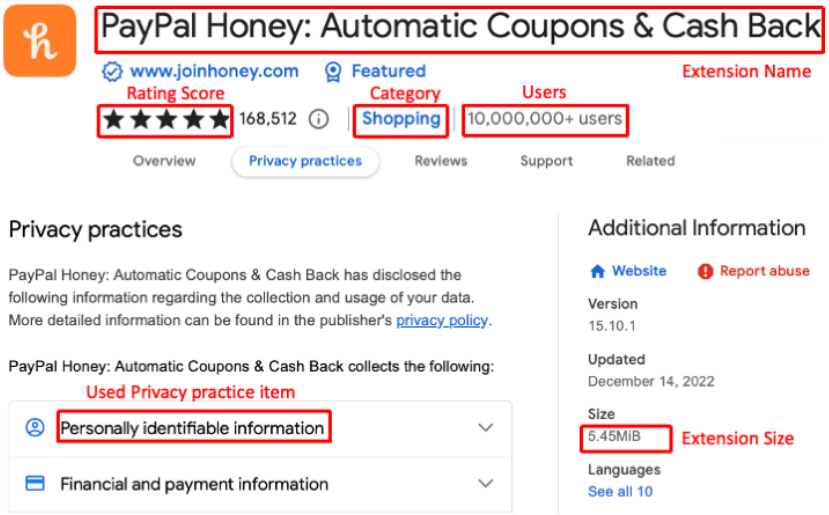

Fig. 2 An annotated screenshot of an extension from the Chrome Web Store.

It is important for users to be aware of the privacy implications of the extensions they install. Thus, Chrome Web Store suggests developers to be transparent about their data collection and use practices by declaring their privacy practices in the Chrome Web Store. Developers document privacy practices to inform users how they collect, use, store, and share their data. Nine types of collected data, including Location, User Activity, Website Content, Personally Identifiable Information, Authentication Information, Web History, Personal Communications, Financial and Payment Information, and Health Information, are documented during the development of extensions and are used to describe privacy practice properties. For example, the use of location services in extensions can track the user's GPS location and is useful for certain types of applications, such as navigation or weather forecasts. User Activity monitors the user's activity when users browse the webpage for the purposes of analytics and personalization.

2.2 Selecting Representative Extensions

The performance behavior of popular extensions may not be generalized to other ones. Thus, we want to extend the existing work (e.g., Pearce, 2020; Merzdovnik et al., 2017; Borgolte and Feamster, 2020), which studies a few selected extensions, with a more diverse set of extensions instead of considering the popularity of extensions solely. In particular, we are interested in investigating all types of extensions in order to generalize the performance impact of extensions on the browser. Due to the enormous number of extensions available, studying all of them is impractical, so we sample extensions to ensure a representative selection. Privacy practices, as a most intuitive factor listed on the Chrome Web Store that could affect browser performance, are considered in our sampling strategy. As such, we use two criteria to select the extensions to ensure their representativeness: (1) we consider the use of privacy practices in extensions to select the studied extensions, as privacy practices (e.g., Location) may have a significant impact on extension performance (Jin, Li, and Y. Zou, 2024; Thiagarajan et al., 2012; Ihara et al., 2015); (2) we include the extensions from different categories. In our work, we cover a wide spectrum of extensions in all 11 categories

of extensions. We cluster the extensions with respect to their privacy practices, and then we select one extension from each category for each resulting cluster.

Clustering extensions based on privacy practices. Although Google technically requires developers to disclose their usage of privacy practices within 30 days [2], but in reality, the policy is not strictly enforced. We observe that the majority (i.e., 60%) of the extensions do not disclose any information about their privacy practices. For instance, the extensions like AlphaText [3] and Justify [4] have not yet provided any details about their privacy practices and are last updated on May 21, 2017, and April 23, 2016, respectively. This is far past the 30-day deactivation period outlined in Chrome Web Store policies as of 2025.Yet, they are still active to use and available to download on the Chrome Web Store, with only a warning. As privacy practices listed on the Chrome Web Store are an intuitive factor that can impact browser performance and the extensions without clarifying privacy practices are untraceable in terms of structure for the analysis, we choose to focus primarily on extensions that adhere to best practices—namely, those disclosing privacy practices. Thus, we exclude such extensions and only consider the extensions with privacy practice specifications to perform the clustering. This also ensures that our measurements are less influenced by ambiguous (undeclared) privacy practices and are replicable for future work. Consequently, this study covers both scenarios—with and without privacy practices. Specifically, we apply the K-medoids clustering algorithm (Schubert and P. J. Rousseeuw, 2021; Schubert and P. J. Rousseeuw, 2019) to cluster the 12,423 extensions with privacy practice specifications. We treat the extensions without privacy practices adopted (i.e., marked as “none” in privacy practices) as a separate cluster (i.e., cluster 0). The K-medoids clustering algorithm outperforms k-means in handling outliers, noise, and non-convex shapes of clusters (Arora, Deepali, and Varshney, 2016; S. Dsouza, J. D. Dsouza, and T, 2017). Since properties in the privacy practices are categorical values, we use one-hot encoding (Hackeling, 2017) to convert the privacy practices of each extension into numerical variables (i.e., 1 means True, and 0 means False). For simplicity, supposing there are three privacy practices: Location ($p1$), Personally Identifiable Information ($p2$), and Authentication Information ($p3$), if an extension uses privacy practices $p1$ and $p3$, the corresponding one-hot encoding is (1, 0, 1). We use the Elbow method (Thorndike, 1953) with silhouette scores (P. J. Rousseeuw, 1987) to choose the number of clusters. The elbow method chooses the “elbow" point in the *Silhouette score ~ the number of clusters curve* (i.e., a relatively small number of clusters with a relatively higher Silhouette score). The optimal K value, i.e., the best number of clusters, is then determined to be 5, with a silhouette score of 0.435. The silhouette score ranges from -1 to 1. The higher value indicates better clustering, and the value of 0.435 suggests a favorable clustering result (P. Rousseeuw, 1987). Adding the cluster of extensions without privacy practices adopted, we obtain a total of 6 clusters, as shown in Table 1.

Selecting representative extensions from each category. It is time-consuming to measure the performance of an extension. To include all 11 categories of extensions in the Chrome Web Store, we select one representative extension from each category in each cluster. The selected representative extension from each category has the minimum Manhattan distance to the medoid of the cluster. The Manhattan distance is a measure of distance between two points in a grid-based system and is calculated as the sum of the absolute differences in their coordinates—one-hot encoding of the privacy practices. A medoid is a representative

[2] Section: "What happens if I don’t fill out the limited use form?" in https://developer.chrome.com/docs/webstore/program-policies/user-data-faq

[3] https://chromewebstore.google.com/detail/lpcaoilgpobajbkiamaojipjddpkkida

[4] (https://chromewebstore.google.com/detail/odkfbmljaomnibofnefflonfehhbhnoo

Table 1 The extension clusters and examples.

Cluster	Explanation and Example	Total Number of Extensions	Number of Extensions Sampled
0	None of privacy practice adopted in the extensions (e.g., Dark Reader[1])	31908	11
1	The cluster focuses on the adoption of website content in privacy practice items (e.g., Wappalyzer[2])	4056	12
2	The cluster focuses on the adoption of authentication information in privacy practice items (e.g., Octotree[3])	1951	11
3	The cluster focuses on the adoption of P.I.I. in privacy practice items (e.g., Keepa[4])	3649	13
4	The cluster focuses on the adoption of user activity and website content in privacy practice items (e.g., TubeBuddy[5])	1586	11
5	The cluster focuses on the adoption of user activity in privacy practice items (e.g., Sourcegraph[6])	1181	14

P.I.I. refers to Personally Identifiable Information
The complete list is put in the replication package.
[1] https://chromewebstore.google.com/detail/eimadpbcbfnmbkopoojfekhnkhdbieeh
[2] https://chromewebstore.google.com/detail/gppongmhjkpfnbhagpmjfkannfbllamg
[3] https://chromewebstore.google.com/detail/bkhaagjahfmjljalopjnoealnfndnagc
[4] https://chromewebstore.google.com/detail/neebplgakaahbhdphmkckjjcegoiijjo
[5] https://chromewebstore.google.com/detail/mhkhmbddkmdggbhaaaodilponhnccicb
[6] https://chromewebstore.google.com/detail/dgjhfomjieaadpoljlnidmbgkdffpack

object of a cluster at the center of the cluster whose the sum of Manhattan distance to all the objects in the same cluster is minimal. The distance is calculated by the similarity of uses of privacy practices between an extension and the medoid. The extensions with the minimum Manhattan distance to the medoid are expected to exhibit the most similar characteristics as the other extensions within the cluster (i.e., being the most representative).

In total, we select 55 extensions from the 5 clusters of extensions with privacy practice specifications (i.e., clusters 1 to 5, listed in Table 1). For the cluster of extensions (i.e., cluster 0, listed in Table 1) that do not adopt any privacy practice (i.e., marked as “none" in privacy practices), the representative extensions cannot be selected in regard to the use of privacy practices, as all these extensions have the same Manhattan distance to each other in terms of privacy practices. Instead, we select representative extensions as to their popularity, i.e., we choose the extensions with the highest number of users in each category. In case two most popular extensions have the same number of users, we consider the rating scores, followed by the number of raters. As a result, we select a total of 11 popular extensions without adopting any privacy practice, choosing one from each category.

There are 6 extensions that are not designated for browsing a website, such as Weather[5]. To improve the representatives of our selected extensions, we retain these 6 extensions and complement them with 6 additional extensions (i.e., ones that work with web browsing) that are closest to the medoid within the same cluster. As a result, a total of 72 representative extensions are selected from the remaining 40% of extensions after excluding the 60% without privacy policy specifications. These comprise 11 extensions that do not have privacy practice specifications to ensure the representativeness of our analysis, 55 extensions that implement privacy practices, and 6 additional complementary extensions. Figure 3 shows the popularity

[5] https://chrome.google.com/webstore/detail/iolcbmjhmpdheggkocibajddahbeiglb

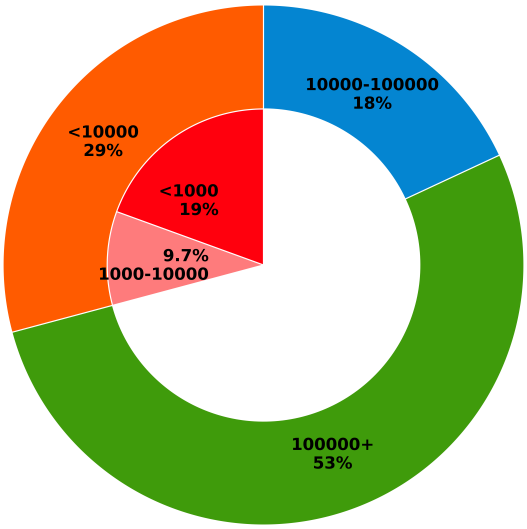

Fig. 3 Extension Popularity.

of the chosen 72 extensions, with 71% of extensions (i.e., 51) having been downloaded over 10,000 times. Table A.1, listed in Appendix A, details the number of downloads for each selected extension. Some representative extensions, such as Tricky Enough, may not have been popular at the time of collection, showing low download numbers, but they could gain popularity over time. For instance, although Tricky Enough has zero downloads when collected, it has 31 downloads and a 4.9-star satisfaction rating as of October 2024. Therefore, we do not filter out extensions with a low number of downloads in order to capture emerging usage patterns of new extensions which may attract more attention over time.

2.3 Designing Testing Scenarios

To measure the performance impact of extensions, we group the extensions based on the similar execution conditions on a website. For instance, extensions Octotree and Sourcegraph share similar working conditions and are intended for use on GitHub, thus we group them together. Afterward, we establish a testing scenario tailored to such extensions. In total, we create seven types of testing scenarios, as listed in Table 2. For instance, 40 extensions are universally compatible (i.e., can be used for all websites) with all types of websites on the internet. The testing scenario for these extensions is named as the `Generic` scenario. The `GitHub` scenario tests extensions that are used to improve the user experience on the GitHub website. We select only code development projects based on the star rankings available on GitHub. For eight extensions that cannot be grouped, we classify their testing scenarios as the `Others` type, comprising of eight individual testing scenarios for the eight extensions. The testing scenarios are summarized in Table 2. For each extension in a scenario, we perform the testing with 10 different websites. For example, we choose the 10 most popular websites from Semrush (2023) as web content for testing the `Generic` scenario and select the 10 best-selling products in each of Amazon (2023)'s top 10 categories for the `Shopping` scenario.

Table 2 Classification of testing scenarios and the corresponding number of extensions.

Scenario	# Ext.	Web content used for testing
Generic	40	10 most visited websites selected from Semrush[1]
Video	10	10 videos that have 720P resolution and last approximately 10 minutes from YouTube or approximately 8.5 hours from Twitch (depending on the designated websites)[2]
Shopping	7	Web pages that host 10 best-selling products selected from each of Amazon's top 10 categories
Sport	3	10 random player profile pages from ESPN[3]
GitHub	2	10 projects that contain around 20 lines of code selected based on the star ranking from GitHub
News	2	10 random press articles: 5 from Naver[4] and 5 from Daum[5]
Others	8	80 designated websites: 10 designated websites of each extension

[1] https://www.semrush.com/blog/most-visited-websites;
[2] We chose specific durations to ensure consistency and make sure they are long enough, especially longer than the measurement period (2 minutes). Twitch, as a live streaming platform, typically has long videos (e.g., 8 hours), while common YouTube users can only upload videos up to 15 minutes by default (Google, n.d.). These durations cover the spectrum of realistic usage scenarios for different users.
[3] https://www.espn.com;
[4] https://news.naver.com; [5] https://news.daum.net.

2.4 Performance Measurement

We study the impact of extensions on the page load time and energy consumption of browsers. More specifically, **the page load time** is the time duration that takes for the webpage to load completely and is measured in seconds. A long page load time (i.e., slow response time) can result in a poor user experience.

We measure two stages of energy consumption as follows:

- **The page load energy consumption** measures the amount of energy in joules (a unit of energy) consumed by the CPU and RAM by the entire system during the page loading time.
- **The stabilized energy consumption** measures the energy consumption of the CPU and RAM by the entire system in joules during a fixed period of time after a webpage has been fully loaded.

To collect energy measurements, we employ Running Average Power Limit (RAPL) (Howard David et al., 2010). RAPL[6] is well-established (H. David et al., 2010; Pereira et al., 2016; Moura et al., 2015) and leverages hardware performance counters to provide detailed and precise reading on system energy consumption of the CPUs and memory usage (Giardino and Ferri, 2016; Khan et al., 2018a). Its accuracy has been validated by various studies (Khan et al., 2018b; Desrochers, Paradis, and Weaver, 2016; Paniego et al., 2018; Kavanagh and Djemame, 2019). The results obtained from RAPL provide a measurement of the total energy consumption in millijoules.

2.5 Testing Procedure

To conduct our experiment, we use a desktop equipped with an Intel i7-4770 @3.4GHz processor, 32 GB of RAM, running Ubuntu (kernel version: 5.15.0-48-generic) with both

[6] https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/advisory-guidance/running-average-power-limit-energy-reporting.html

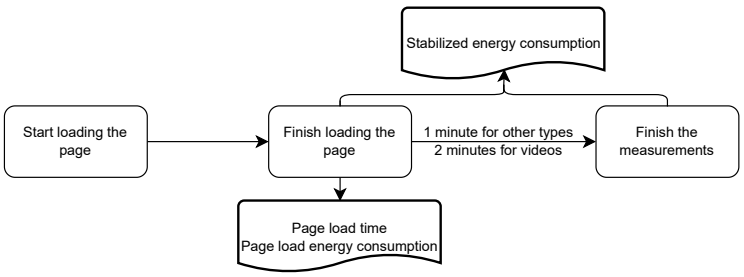

Fig. 4 Measurement timeline

WiFi and Bluetooth disabled. We use Selenium[7], a tool that generates simulation scripts, to automatically execute a web browser to access designated websites. The Google Chrome browser of version 104.0.5112.79 (Official Build 64-bit) is utilized in our experiment.

To account for measurement errors, we calculate the minimal number of repetitive measurements for each performance metric to achieve an acceptable level of accuracy, using Equation 1 (Jain, 1991).

$$\mathrm{n} = \lceil (\frac{100 \times z_{1-\alpha/2} \times s}{r \times \bar{x}})^2 \rceil \tag{1}$$

where n stands for the number of observations required; $z_{1-\alpha/2}$ is the $1-\alpha/2$ critical value of the normal variate at the desired confidence level $(1-\alpha)$; s represents the median standard deviation of the measurements for each website and extension combination; r denotes the required accuracy; and $\bar{x}$ represents the median value of the sample mean of the measurements for each website and extension combination.

We calculate the median standard deviation (s) and the median sample mean ($\bar{x}$) of performance metrics for each extension. 100 measurements of an extension are divided into 10 sets, each corresponding to a specific website. For each website, we compute the mean and standard deviation of the 10 measurements. We then take the median of these 10 means to determine $\bar{x}$ and the median of the 10 standard deviations to determine s. With a 95% confidence level and a required accuracy of 10%, we obtain the minimal number of repetitive measurements for the performance metrics based on Equation (1): 9 times for the page load time, 6 times for the page load energy, and 1 time for the stabilized energy consumption. The different numbers of repetitive measurements stem from the inherent characteristics of each metric and the impact of measurement errors. For instance, page load time might exhibit more variability, necessitating a larger number of measurements to achieve the desired accuracy level. Stabilized energy consumption, being more stable, requires fewer measurements to meet the accuracy criteria. In our experiment, we choose an ample number of repetitive measurements (i.e., 10) that is larger than the largest number required for any of the performance metrics (i.e., 9).

We use scripts to open and navigate to the designated web pages and simulate users' activities in practice. To measure each testing scenario, we perform the experiment in the following steps:

1) Utilize Selenium to launch the Google Chrome browser.

2) Install and configure an extension on demand.

3) Repeat each test case (i.e., each website under each scenario) 10 times to obtain stable measurements in that testing scenario, as specific energy measurements may vary across executions of the same testing scenario. For each run of a testing scenario, the local storage

[7] https://www.selenium.dev

and caches of the browser are cleaned to avoid that the caches would maintain the results of the previous experiments. In total, we test and measure the performance of each extension 100 times (i.e., 10 websites, each with 10 repetitions (Georgiou et al., 2022)).

4) Collect performance measurements of energy and running time while the testing scenario is running, as illustrated in Figure 4.

5) Sleep for one minute to avoid tail power states (Bornholt, Mytkowicz, and McKinley, 2012) to allow the system to reach a stable condition again (idle energy consumption) before executing the next run (i.e., running the browser to access one website).

6) Terminate the Google Chrome browser for each finished task by closing all windows and pages, thereby ending the browser session that is opened during measurement.

7) Uninstall the current extension.

8) Repeat 1) to 7) for another extension.

9) Collect the energy consumption of the CPU and RAM as well as the page load time on each website without any extension.

The page load time and the page load energy consumption are monitored from the time the webpage starts loading until the webpage completes the loading. Once the page finishes loading, the stabilized energy consumption of CPU and memory usage is measured for one minute for non-video testing scenarios and two minutes for the `video` testing scenario. We choose the one-minute testing time to reach a balance between the duration of the measurement and the time resources available for executing all the testing scenarios (it takes approximately 10 hours to execute the testing scenarios). The two-minute measurement duration for the `video` scenario is intended to account for the ad time in videos.

By repeating each experiment 10 times, we obtain stable measurement results across the page load time, page load energy consumption, and stabilized energy consumption. These measurement results exhibit good stability, as evidenced by median standard errors (Curran-Everett, 2008) of 0.050, 0.042, and 0.0035 (calculated from measured metric values normalized by median normalization in Section 2.6) for the respective metrics. In the experiment, Selenium initially opens a blank webpage to avoid caches and the impact from previous runs on the measurements, before the initial page is directed to the testing webpage.

2.6 Normalization

Normalization ensures that measurements are comparable across different websites and testing scenarios by standardizing them on a consistent scale. For example, normalization helps mitigate the confounding impact caused by different websites' performance differences when assessing extensions' performance impact in RQ1 and RQ2. For RQ3, we require normalization to avoid raw values distorting the model's results. Consequently, to ensure consistency and comparability of all measurements collected during the testing of extensions, regardless of the testing scenarios and the websites, we normalize the measurements obtained with and without the use of an extension. For each extension, the normalized value of the jth ($j \in 1, 2, .., 10$) measurement of the browser's performance when accessing the ith ($i \in 1, 2, .., 10$) website (perf_norm_{ij}) is calculated by Equation (2).

$$\text{perf\_norm}_{ij} = \frac{\text{perf\_ext}_{ij}}{\text{median}(\text{perf\_free}_i)} \tag{2}$$

where perf_ext_{ij} is the raw value obtained by the jth measurement of the browser performance while the extension accesses the ith website; and $\text{median}(\text{perf\_free}_i)$ is the median value obtained by the 10 measurements of the browser performance (free of extension) when

the extension accesses the *i*th website. Similarly, the performance measures of the browser in the extension-free mode are normalized by the median value of the 10 measurements for each website.

3 Experiment Results

The objective of this study is to investigate the impact of extensions on the energy consumption and page load time of browsers through a case study on Google Chrome. This section describes our research questions, our approaches, and answers to each of them.

3.1 RQ1: How do extensions impact browser performance?

Motivation: Extensions are developed to enrich the browser functionality, such as easy dictionary searching or blocking advertisements. However, practitioners may not be aware that extensions may cause significant performance impact on browsers, such as extra CPU computation and memory usage, and increased the page loading time. Adversely, extensions could result in degraded user experience and impaired performance of other applications running on the same computer. In this RQ, we strive for understanding the extent to which extensions affect the overall browser performance.

Approach: We conduct performance evaluations of 72 representative extensions by executing their corresponding testing scenarios (Section 2.3). For each extension, we execute the designated websites 10 times with and without the extension installed (Section 2.5). Other extensions are uninstalled. Measurements are then collected and normalized, as described in Section 2.6. Subsequently, we perform statistical analysis to understand the performance impact of the extensions.

Configuring the extensions. Different configurations of an extension may have different performance impacts[8]. In this RQ, we consider the expected situation when an extension is used in the fully-loaded mode: the extension is activated and used for the designated websites; for extensions that require access and login, we grant the extension access and logs the extension in. As the baseline, we also measure the performance of the browser when the extensions are uninstalled (i.e., extension-free mode).

Correlation analysis. To understand the relationship among the different performance metrics, we use the Spearman rank ρ correlation analysis on each pair of performance metrics to identify the degree of correlation. The reason behind using the Spearman Rank Correlation is that it does not prerequisite a normal distribution. The Spearman Rank Correlation ranges from -1 to +1, where +1 indicates a perfect positive association between ranks, 0 means no association, and -1 represents a perfect negative association.

Statistical analysis. To quantify how extensions affect the browser performance, we test the following hypothesis:

$H0_1$: *there is no difference in the distributions of performance metric values between the paired observations of extension-free and fully-loaded modes.*

To compare the two distributions of performance metric values (i.e., fully-loaded mode vs. extension-free mode) for each extension, we apply the Mann-Whitney U test (Mann and Whitney, 1947) at a 5% significance level to assess whether a statistically significant

[8] The performance impact of the extensions' different configurations (i.e., usage modes) is discussed in detail in RQ2.

difference exists. The Mann-Whitney U test, as a non-parametric statistical test, does not assume a normal distribution. If $H0_1$ is rejected (i.e., a statistically significant difference exists), we further compute the Cliff's δ effect size (a non-parametric method without assumption of a particular distribution) (Cliff, 1993), which quantifies the magnitude of the differences (J. Romano et al., 2006). The resulting effect sizes are classified into several qualitative degrees of difference (Benjamini and Hochberg, 1995): negligible ($|\delta| < 0.147$), small ($0.147 \leq |\delta| < 0.33$), medium ($0.33 \leq |\delta| < 0.474$), and large ($|\delta| \geq 0.474$). A larger effect size signifies a larger difference between the two distributions. Following prior studies (Tong, Liu, and S. Wang, 2018; Aniche et al., 2016; D. Romano et al., 2012), we choose the small effect size as the threshold for determining significant differences in the performance distributions: the performance comparisons resulting in a negligible effect size are not considered significant differences.

Performance change ratio. If $H0_1$ is rejected, we calculate the change ratio of the performance metric as specified in Equation (3). To reduce the effect of outliers in website measurements, the median value of each 10 repeated experiments is used.

$$Ratio(metric) = \frac{1}{10}\sum_{i=1}^{10}\frac{median(ext_i) - median(free_i)}{median(free_i)} \tag{3}$$

where $Ratio(metric)$ is the change ratio of the corresponding performance metric of an extension (i.e., the page load time, the page load energy consumption, or the stabilized energy consumption); ext_i is the normalized performance metric under the fully-loaded mode of the extension; $free_i$ is the normalized performance metric under the extension-free mode of the extension; $median(ext_i)$ and $median(free_i)$ indicate the median of the 10 extension measurements for the i-th website, under the fully-loaded mode and the extension-free mode, respectively. Finally, $i \in 1, 2, .., 10$ indicates the i-th tested website for the extension. The mean is typically better when the data follow a symmetric distribution. However, our measurement results exhibit a high variability across different websites. Therefore, we use the median of measurements in our work, as it is less sensitive to outliers and better captures the central tendency in performance metrics, ensuring that extreme values from specific websites or test runs do not skew the results.

Findings: Using an extension can either deteriorate or improve browser performance, while performance deterioration (especially large deterioration) is more common. Table 3 presents the statistically significant changes in the performance metrics when an extension is used in its fully-loaded mode. A total of 72 extensions are tested, among which 66 extensions (i.e., 92%) exhibit a statistically significant performance impact on at least one of the studied performance metrics. In particular, 40 extensions (i.e., 56%) exhibit statistically significant changes in the **page load time**, among which 22 increase the metric value while the other 18 decrease the metric value. 46 extensions (i.e., 64%) exhibit statistically significant changes in the **page load energy consumption**, among which 43 and 3 lead to an increase and a decrease in the metric value, respectively. 53 extensions (i.e., 74%) exhibit statistically significant changes in the **stabilized energy consumption**, with 47 and 6 of them increasing and decreasing in the metric value, respectively. **In particular, performance deterioration dominates the biggest performance changes (i.e., with *large* effect sizes).** For example, 23 extensions lead to *large* increases, but 0 extension decreases in stabilized energy consumption.

The deterioration in browser performance may be attributed to the extra resources required to run the extensions, the automatic backend search capabilities of certain extensions, such as potential coupon detection (e.g., SimplyCodes) and tracking of product price history (e.g., Keepa), web content adjustments (e.g., Octotree), monitoring of website content (e.g.,

Table 3 Statistical performance changes caused by the extensions in their fully-loaded mode.

Statistical analysis		Mann-Whitney		Cliff's δ					
		Signif.		Small		Medium		Large	
metrics	Tendency	Count	Ratio	Count	Ratio	Count	Ratio	Count	Ratio
Page load time	Increase	22	18%	11	14%	3	29%	8	19%
	Decrease	18	-6.3%	11	-4.2%	5	-9.0%	2	-43%
	Overall	40	4.0%	22	0.48%	8	-4.6%	10	18%
Page load energy consumption	Increase	43	17%	14	9.0%	8	18%	21	35%
	Decrease	3	-7.3%	0	-	2	-6.6%	1	-61%
	Overall	46	16%	14	9.0%	10	15%	22	31%
Stabilized energy consumption	Increase	47	1.8%	14	1.4%	10	1.3%	23	3.0%
	Decrease	6	-1.6%	4	-0.95%	2	-3.0%	0	-
	Overall	53	1.6%	18	1.0%	12	0.79%	23	3.0%
# of extensions being tested		72							

* This table shows the number of extensions that have a statistically significant effect on each performance metric, including an increase, decrease, and overall change (increase and decrease), as well as the respective change ratio. The Signif. column counts extensions with a p-value less than or equal to 0.05. The change ratios are the mean values of the extensions.
* The results of the effect sizes, as determined by Cliff's δ, are categorized as Negligible (not reported in the table), Small, Medium, and Large. The effect sizes are calculated and reported only when the performance impact is statistically significant, i.e., the p-value is less than or equal to 0.05 in the Mann-Whitney U test.

Dynatrace), and site analysis (e.g., Wappalyzer) during page load. On the other hand, the improvement in the browser performance may be due to the blocking of web content (i.e., Naver/Daum Media Filter and ad blocker: Inforness), simplification of web content (e.g., Better Tab), and preventing information trackers (e.g., Neeva).

Page load time is strongly correlated with page load energy consumption ($|\rho|$=0.82 > 0.70) but has a negligible correlation with stabilized energy consumption (close to 0) (Schober, Boer, and Schwarte, 2018), as shown in Figure 5. This is because both page load time and page load energy consumption relate to the loading activity, while stabilized energy consumption occurs after loading, thus it is not directly associated with the other two metrics.

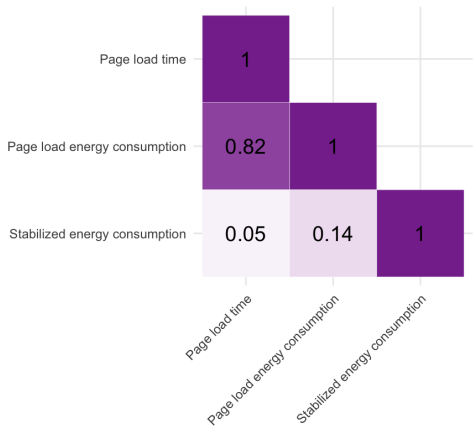

Fig. 5 The results of Spearman correlations between performance metrics.

22 (31%) extensions statistically significantly increase the page load time, with an average increase of 18%, while another 18 (25%) extensions statistically significantly reduce the page load time by 6.3% on average. Over half (40/72) of the extensions lead

to statistically significant changes in the page load time, with an average change of +4.0% (increase). In particular, eight extensions lead to an increase in the page load time by a large effect size, while only two extensions lead to a decrease by a large effect size. Web browsers or extensions may provide warnings about the extensions' impact on the page load time when users install such extensions, suggest users to disable certain extensions when accessing performance-critical websites, or optimize the extension or browser-extension integration to minimize the negative performance impact.

Using extensions leads to the largest negative impact on the page load energy consumption, compared to other performance metrics. 46 out of the 72 extensions demonstrate statistically significant differences in the page load energy consumption, with an average change of 16% (increase). A statistically significant increase in energy consumption is observed in 60% of the extensions (i.e. 43), with an extra 17% of energy on average consumed. Conversely, only 4.2% of the extensions (i.e., 3) reduce energy consumption, by 7.3% on average. In particular, 21 (29%) extensions exhibit a large effect size in the increase of the page load energy consumption, by an average increase of 35%, while only one extension decreases the page load energy consumption by a large effect size. Browser developers may provide more information about the energy consumption of the browser and the extensions to help users make informed decisions when using an extension and to motivate extension developers to improve the energy efficiency of their extensions.

Even after the web page has been loaded, 47 (65%) extensions still lead to a statistically significant increase in energy consumption. The majority of extensions, 74% (i.e., 53 extensions), demonstrate a statistically significant difference in the stabilized energy consumption. 65% (i.e., 47) of the extensions consume an extra 1.8% of the stabilized energy on average, while 8.3% (i.e., 6) of the extensions reduce the stabilized energy consumption by an average of 1.6%. In particular, 32% (i.e., 23) of extensions demonstrate large differences in the stabilized energy consumption, with an energy difference of 3.0%. Compared to the impact on the load time energy consumption, the extensions' impact on the stabilized energy consumption is relatively small. This may be explained by the fact that the extensions are less active after the pages have been loaded.

Summary: Our results indicate that the use of extensions under the fully-loaded mode can lead to a statistically significant impact on the browser performance, with the largest negative impact on the load time energy consumption. Our observations suggest that browser and extension developers should pay attention to the performance impact of browser extensions. For example, they may provide warnings and information about the performance impact of the extensions, suggest users disable extensions in performance-critical scenarios and optimize the extensions or browser-extension integration in terms of performance.

3.2 RQ2: How do the usage modes of extensions affect browser performance?

Motivation: In RQ1, we have analyzed the effect of the extensions on the browser performance in the expected scenario (i.e., the fully-loaded mode). Nonetheless, users may not always interact with extensions in the intended manner. For instance, a user may ignore the login process of an extension to utilize the extension. The impact of the extensions under different usage modes on browser performance remains uncertain. Therefore, it is important to understand how different usage modes of extensions could lead to varying performance impacts and to provide insights for browser users to select and configure extensions.

Approach: The usage level of an extension depends on how the user configures and uses it (e.g., whether the extension is logged in or used with the designated websites). We first provide

a classification of the different usage modes of the extensions, then perform experiments to test the performance impact of the different usage modes.

Classifying and testing the usage modes of extensions. Based on the requirements for extension access and execution (e.g., requirements for login or getting access to a website), as well as their intended usage scenarios (e.g., used with intended or designated websites), we classify usage modes of extensions into six categories: *extension-free* (the baseline), *fully-loaded*, *no-login*, *no-grant*, *non-designated*, and *fully-inactive*, as outlined in Table 4. Browsing web pages without the extension installed is the extension-free mode, which serves as the baseline for performance comparison. The usage mode of extensions can be classified in four modes, considering whether the extension is logged in, granted access to the website, tested with the designated webpages, and enabled. The description of different usage modes is listed in Table 4. Except for the *fully-loaded* mode, not all extensions necessitate certain conditions (e.g., login): usage modes of extensions are applicable only to partial extensions. Specifically, all the 72 extensions can be tested in the *fully-loaded* mode. 13 extensions can be executed without login (i.e., *no-login* mode), and 17 extensions are tested in both the websites that are not designated for the extensions (*non-designated* mode) and in the websites such that the permission to access a webpage is not granted (i.e., *fully-inactive* mode), respectively.

We evaluate the performance impact of the different usage modes by comparing the measures of extensions in partial usage modes with both the baseline (the extension-free mode) and the measurement results of extensions in the fully-loaded mode. All 72 extensions are selected for testing under the fully-loaded mode (same as in RQ1). We select the extensions to be tested in the partially loaded modes based on their requirements and usage scenarios. We test the extensions that require access permission to the tested webpages in the no-grant mode. We test the extensions that require login in the no-login mode. The non-designated mode involves testing the inactive extensions that only work for designated websites and is tested with the context of generic testing scenario websites. The performance of the inactive extensions is assessed in the fully-inactive mode which does not log in to the extension, not grant access to the webpage, and is tested with the generic testing scenario websites rather than the designated websites to the extensions that only work for the designated websites. We measure a total of 40 extensions across the partial usage modes. Tables 5 and 6 list the distribution of the number of extensions tested in each usage mode. As shown in Tables 5 and 6, we can compare the extension-free mode with the fully-loaded mode. We follow the same measurement procedures, as outlined in Section 2.5. The measurements are recorded and then normalized using the procedure outlined in Section 2.6.

Statistical analysis of the experiment results. Similar to RQ1, we apply the Mann-Whitney U test and the Cliff's δ test to determine the degree of difference of the measurements between running a usage mode of an extension and running in the extension-free, as well as between running a usage mode of an extension and the fully-loaded mode. We test the following two hypotheses:

$H0_2$: *there is no difference in the distributions of performance metric values between the paired observations in a partial usage mode and the* ***extension-free mode****.*

$H0_3$: *there is no difference in the distributions of performance metric values between the paired observations in a partial usage mode and the* ***fully-loaded mode****.*

Performance change ratio. When an extension leads to a statistically significant performance difference between the performance metrics under a usage mode and the extension-free mode or the fully-loaded mode, we calculate the change ratio of the performance metric, following the same method described in RQ1.

Table 4 Usage modes of extensions.

Mode name	Login*	Grant*	Designated webpage**	Extension installed	Extension enabled	Description
Extension-free	0	0	0	0	-	No extensions are installed. **(Baseline)**
Fully-loaded	1	1	1	1	1	Extension is logged in; extension is tested with the designated webpages; extension's access permission to the webpages is granted **(Baseline)**
No-login	0	1	1	1	1	An extension is not logged in
No-grant	1	0	1	1	1	Extension's access permission to the tested webpages is not granted
Non-designated	1	1	0	1	0 ***	An extension is not tested with the designated webpages
Fully-inactive	0	0	0	1	0	Access to an extension and access permission to a webpage are not granted; not tested with the designated webpages

1 represents True (i.e., performed); 0 represents False (i.e., not performed); "-" represents not applicable.
* If an extension does not need to be logged in or to be granted access, it is always considered as being logged in or granted access.
** A designated webpage of an extension is the webpage that an extension is designed for (e.g., GitHub); the generic extensions (corresponding to the generic scenario in Table 2) have all webpages as their designated ones. Extensions with designated websites are designed to be used only on specific sites but not entirely disabled elsewhere. For instance, such extensions may need to monitor whether the user switches to their designated page, which can lead to increased energy consumption.
*** Extensions are naturally disabled when accessing non-designated webpages.

Table 5 Comparing the performance impact of different usage modes of the extensions with the extension-free mode.

Mode		No-grant		No-login		Non-designated		Fully-inactive	
Performance metrics	Tendency	Count	Ratio	Count	Ratio	Count	Ratio	Count	Ratio
Page load time	Increase	2	43%	4	40%	4	20%	6	110%
	Decrease	4	-6.9%	2	-4.8%	0	-	5	-32%
	Overall	6	-6.3%	6	25%	4	20%	11	6.3%
Page load energy consumption	Increase	8	7.5%	14	18%	5	14%	5	280%
	Decrease	0	-	0	-	0	-	3	-29%
	Overall	8	7.5%	14	18%	5	14%	8	8.7%
Stabilized energy consumption	Increase	12	2.8%	15	2.9%	3	1.6%	8	13%
	Decrease	0	-	0	-	6	-0.78%	3	-28%
	Overall	12	2.8%	15	2.9%	9	-0.43%	11	8.8%
# of extensions being tested		13		17		11		11	

We share our Cliff's delta results in our replication package.

Findings: Browser performance is impacted by the use of extensions regardless of their usage modes. The results of testing $H0_2$ and $H0_3$ are presented in Tables 5 and 6, indicating the relative performance impact of the extensions in their four partial usage modes, namely: no-grant, no-login, non-designated, and fully-inactive, in comparison to the two baselines (the extension-free mode and the fully-loaded mode). Table 5 shows that all partial usage modes of the extensions can statistically significantly impact the browser performance in terms of the studied performance metrics. Overall, 12 out of the 13 extensions (i.e., 92%) tested with the no-grant mode, 16 out of the 17 extensions (i.e., 94%) tested with the no-login mode, 10 out of the 11 (i.e., 91%) extensions tested with the non-designated mode, and 11 out of the 11 extensions (i.e., 100%) tested with the full-inactive mode statistically significantly impact at least one of the studied performance metrics.

Table 6 Comparing the performance impact of different usage modes of the extensions with the fully-loaded mode.

Mode		No-grant		No-login		Non-designated		Fully-inactive	
Performance metrics	Tendency	Count	Ratio	Count	Ratio	Count	Ratio	Count	Ratio
Page load time	Increase	1	15%	5	14%	7	25%	4	200%
	Decrease	2	-6.8%	2	-8.3%	3	-6.5%	4	-36%
	Overall	3	-3.6%	7	9.8%	10	8.4%	8	46%
Page load energy consumption	Increase	2	8.3%	6	6.5%	3	190%	6	220%
	Decrease	1	-3.0%	2	-5.4%	6	-4%	3	-39%
	Overall	3	1.2%	8	5.0%	9	2.6%	9	2.9%
Stabilized energy consumption	Increase	8	2.6%	11	3.2%	3	1.6%	7	15%
	Decrease	3	-2.3%	1	-0.92%	5	-4.9%	3	-38%
	Overall	11	1.6%	12	3.2%	8	1.3%	10	11%
# of extensions being tested		13		17		11		11	

We share our Cliff's delta results in our replication package.

Using extensions in unexpected circumstances (i.e., when access to a webpage is not granted or an extension is not logged in) can still lead to impaired browser performance, especially more energy consumption. For example, 62% of the tested extensions (i.e., 8) running without a granted permission (i.e., no-grant mode) and 82% of the extensions (i.e., 14) tested without the required login (i.e., no-login mode) lead to an average of 7.5% and 18% increase in the page load energy consumption, respectively. 88% of extensions (i.e., 15) in the no-login mode and 92% of extensions (i.e., 12) in the no-grant mode result in a slight increase in the stabilized energy consumption, leading to an average of 2.8% and 2.9% increase, respectively. When users opt to disregard the login process of the extensions, it is observed that approximately 35% of the tested extensions (i.e., 6) significantly contribute to the page load time, leading to an average delay of 25% on average. In comparison, denying access permission to the website is more likely to decrease the response time while leading to an increase in energy consumption during the page load and the stabilized periods. In particular, 46% of the extensions (i.e., 6) significantly impact the page load time, resulting in an average of 6.3% decrease on average. Under unexpected circumstances, extensions may not be able to utilize their full functionality to efficiently load and interact with web content, leading to suboptimal performance and potential disruptions during browsing activities.

Even when extensions are not used for their designated websites (e.g., TubeBuddy for `Videos`) or are fully deactivated, they can still lead to significant energy consumption and increase the page load time. In particular, 8 to 11 (73% to 100%) of the extensions tested under the fully-inactive mode significantly impact the studied performance metrics. Such observation may be attributed to the fact that extensions may still run background processes and built-in functionalities or scripts, consuming resources even when they are not actively being used. The non-designated mode results in statistically significant changes on 5 extensions (45%) in the page load energy consumption by an average of 14% on average. The negative performance impact of the non-designated modes may be because the extensions make attempts to access the web content when a webpage is loading, which may cause significant extra overhead.

The partial usage modes of extensions can lead to even worse performance impact than the fully-loaded mode. Table 6 indicates the relative performance impact of the different usage modes in comparison to the fully-loaded mode. Surprisingly, all partial usage modes can lead to a worse performance impact even than the fully-loaded mode. For example, 8 out of the 17 extensions (i.e., 47%) tested under the no-login mode exhibit a statistically significant increase in the page load energy consumption in comparison to the fully-loaded

mode, leading to an average increase of 5.0% in the metric value. 8 out of the 11 extensions (i.e., 73%) tested under the fully-inactive mode significantly impact the page load time over the fully-loaded mode, with an average increase of 46%. When extensions are not used in the intended mode (i.e., fully-loaded), they still run and consume resources (e.g., through continuous attempts of obtaining access), which may consume more resources and lead to worse user experience than the fully-loaded mode. The surprising results indicate the need for better performance testing of the extensions under the unintended usage scenarios.

Summary: We observe that browser performance can be negatively impacted by the use of extensions even when they are used in unexpected circumstances (e.g., not logged in or access to a webpage not granted) or are not active (e.g., not used for designated websites). Surprisingly, unintended usage scenarios of extensions can even lead to worse performance impact than the fully-loaded mode. Extension users should be aware of such performance impact to optimize their configurations of the extensions (e.g., avoiding improper use). Our findings also suggest that browser and extension developers should take action on reducing the performance impact of extensions under unintended usage scenarios (e.g., better performance testing and optimization for such scenarios).

3.3 RQ3: What factors of extensions influence browser performance?

Motivation: Browser extensions are designed with various characteristics, such as the adopted privacy practices and usage modes to provide users with various functionalities. In RQ2, we have observed that different usage modes of extensions lead to different impacts on the browser performance. In this RQ, we are interested in understanding multifaceted factors that can significantly influence browser performance. We want to unravel the interplay of these factors. With such insights, extension developers can use the knowledge of performance influencing factors to optimize their extensions. Moreover, extension users can make more informed decisions to select extensions that best meet their needs by considering various factors that may potentially affect the browser performance.

Approach: We extract a set of factors (e.g., code metrics) of the extensions and leverage a statistical model to understand the influential factors.

Extracting extension factors. Table 7 lists 105 considered factors of the extensions. We consider the common factors that might impact performance from five aspects, including code metrics, file characteristics, privacy practices, user perspectives, and usage modes of extensions. We employ SciTools' Understand[9] to analyze the code metrics of the extensions, e.g., Lines of Code (LOC) and Number of Children (NOC). The Understand tool evaluates the code using various code metrics[10] and also provides entity information, such as the number of classes and public methods utilized in the extension. We consider the code metrics proposed by Chidamber and Kemerer (1994) and Lorenz and Kidd (1995). Beyond the code metrics and entities, we take into account the types of collected data and file characteristics (e.g., the file size in different kinds of files related to the extension, such as the size of .png typed files). 72 compressed extension projects (i.e., .crx files) are extracted via the online web tool[11] to obtain the source code. Besides code metrics, we complement additional factors as per the inherent nature of the extensions, including the belonged categories, user perspective, and usage scenarios (i.e., usage modes). A total of 105 potential factors of the extensions that may impact the browser performance are listed in Table 7.

[9] https://scitools.com/

[10] https://documentation.scitools.com/pdf/metricsdoc.pdf

[11] https://crxextractor.com/

To understand the interplay of these factors on the browser performance, we construct a statistical model. The dataset is collected from the same experiment performed in Section 3.2, and the characteristics of the extensions (e.g., extension category) are collected based on the steps described in Section 2.1.

Table 7: Studied factors that may influence the performance impact of browser extensions.

Variable Type	Potential Influential Factors	Description
Code Metric	Ambient Module	Number of ambient modules
	Class	Number of classes
	ClassFunction	Number of class functions
	Enum	Number of enum classes
	File	Number of files
	Interface	Number of interfaces
	Method	Number of methods
	Namespace	Number of namespaces
	Private Method	Number of private methods
	Public Method	Number of public methods
	Public Static Method	Number of public static methods
	Function	Number of functions
	Unnamed Function	Number of unnamed functions
	WMC (Weighted Methods per Class)	Number of local (not inherited) methods
	DIT (Depth of Inheritance Tree)	Maximum depth of class in inheritance tree
	ev(G) (Essential complexity)	The number of decision points + 1 after control graph reduction
	CC (Cyclomatic Complexity)	The number of decision points + 1
	NPATH (Number of Possible Paths)	Number of unique paths trhough a body of code, not countingabnormal exits or gotos
	CLOC (Comment Lines of Code)	Number of lines containing comment
	LOC (Lines of Code)	Number of lines containing source code
	BLOC (Blank Lines of Code)	Number of blank lines
	NL (Number of Lines)	Number of Lines
	NPM (Number of Public Methods)	Number of local (not inherited) public methods
	NPRM (Number Private Methods)	Number of local (not inherited) private methods
	RFC (Response for a Class)	Number of methods, including inherited ones
	NIV (Number of Instance Variables)	Number of instance variables
	NIM (Number of Instance Methods)	Number of instance methods
	NV (Number of Variables)	Number of class variables
	NOC (Number of Children)	Number of immediate subclasses
	IFANIN	Number of immediate base classes
File Characteristic	Sizes of different type of files	Size of various types of files in the extension
Privacy Practice	Total # of Privacy Practices Used	Number of privacy practice properties adopted by the extension
	Location User Activity Website Content	Seven privacy practice items adopted by the extension

Continued on next page

Table 7 – continued from previous page

Variable Type	Potential Influential Factors	Description
	Web History P.I.I. Authentication Information Personal Communications	
User Perspective	Number of Users Number of Raters Extension Size Rating Score	Number of users installing the extension Number of users rating the extension The extension package size The rating score of the extension in Chrome Web Store
Usage Mode	Logged in Access to webpage granted Non-designated mode Fully-inactive mode	Four usage modes of extensions

P.I.I. refers to Personally Identifiable Information
The complete list is put in the replication package.

Redundancy analysis. Variable selection is incorporated into the model-building procedure to aid in raising the accuracy (Giglio and Brown, 2018; H. Zou and Hastie, 2005), whereas redundancy is not considered, as redundant predictors unnecessarily deteriorate the performance of the regression models (Xue et al., 2016; Lin et al., 2015). To uphold model robustness and minimize complexity, we conduct a redundancy analysis before building models to avoid redundant factors that could interfere with each other using the `redun` function from the `Hmisc` library in R. To detect if a variable is redundant, a multivariate regression model is constructed for each explanatory variable, treating it as the response variable and using other variables as explanatory variables. The fit of these models is measured using R^2 statistics. A high R^2 value indicates redundancy, as it shows how well each explanatory variable is explained by other variables, thereby helping to identify which variables are redundant. This analysis assesses the redundancy correlation among the 115 identified factors. Factors that can be explained by other factors with an R^2 larger than 0.90 are deemed highly redundant and thus eliminated from the result set. As a result, 24 factors (i.e., *Ambient Module*, *Interface*, *Namespace*, *Private Method*, *Enum*, *Class*, *Class Function*, *Public Method*, *Unnamed Function*, *Method*, *File*, *NIM*, *NPRM*, *NPM*, *DIT*, *RFC*, *ev(G)*, *NIV*, *NL*, *CC*, *IFANIN*, *WMC*, *CLOC*, and *BLOC*) among the code metrics, 40 factors (i.e., *Size of .mem, .data, .psd, .sendkeys, .datepick, .tz, .blockUI, .dat, .vtt, .otf, .mjs, .1, .6, .ts, .patch, .targ, .lock, .opts, .cjs, .in, .def, .jst, .coffee, .htm, .zip, .avif, .config, .md, .mp4, .conf, .sortable, .webm, .js, .woff2, .html, .txt, .gif, .woff, .ico, .css, and .xml files*) among the file characteristics, *extension size*, and *total number of privacy practices used* are redundant and not considered in our model. In the end, we keep 37 factors after the redundancy analysis for inclusion in our model.

Model Construction and Evaluation. To study the interplay of the factors on performance metrics (e.g., the page load time), we build elastic net regression models (Friedman, R. Tibshirani, and Hastie, 2010; Simon et al., 2011; Tay, Narasimhan, and Hastie, 2023) using the extension factors as explanatory variables and the performance metrics as response variables. We use the `train` function (Kuhn, 2008) provided by the `caret` library in R to construct elastic net regression models for each performance metric with the gaussian method. We normalize each factor by dividing each value by the largest value in the dataset, which facilitates the convergence of coefficients.

In the quest for finely tuned hyperparameters, we utilize the `trainControl` (Efron and R. J. Tibshirani, 1994; Efron, 1983; Kuhn, 2014) from the `caret` package. We opt for adaptive resampling (`adaptive_cv`), employing adaptive cross-validation for a total of 100 times parameter tuning. We assess each performance metric with regard to extension factors in the tuning process. This process leverages the Bradly-Terry resampling method, which manages a substantial number of tuning parameter settings, incorporated with a random search (Bergstra and Bengio, 2012) to ensure comprehensive coverage. To understand the fitness of the tuned model, we explore a range of tuning parameter combinations (i.e., `tuneLength`) through a random search. In essence, `tuneLength` signifies the scope or breadth of the search space for optimal tuning parameters. We systematically vary the `tuneLength` within the range of 1 to 300, allowing for a comprehensive evaluation of model fitness across different parameter settings. The criteria for selecting the optimal model hinges on the root-mean-square error (RMSE), which is also chosen by the function selection in fine-tuning process by default. The model with the lowest RMSE is targeted. RMSE is used to select the optimal model using the smallest value. As a result, our elastic net regression models illustrate an RMSE of 0.020 for the page load time, 0.043 for the page load energy consumption, and 0.067 for the stabilized energy consumption, respectively. Given that the metrics are normalized between 0 and 1, these RMSE values are relatively low, indicating that the model performs well in capturing the underlying patterns in the data, which supports the fitness of our models.

Analyzing the influences of the factors. Features with a non-zero coefficients are significant and relevant for the model, as the non-significant ones are already pruned by the lasso penalty. We determine the effect direction of a significant factor to indicate whether a significant factor has a positive or negative impact on a performance metric. The effect direction of a significant factor is in fact the sign (+ or −) of the coefficient of the significant factor in the elastic net regression models.

Findings: Our model analysis comports with the observations in RQ2, confirming that the usage modes of extensions significantly affect the browser performance. Logged-in extensions tend to have a faster page load time and lower energy consumption. The negative relationships between *Logged in* and the performance metrics suggest that logging in to the extension can improve browser performance. Therefore, it is recommended that users log in to the extension when prompted to do so. Besides, it is advisable for users to grant permission to the extension to access the webpage, as indicated by the negative correlation shown in Table 8 - Usage Mode. The negative correlation suggests that granting permission (i.e., *Access to webpage granted*) leads to an improvement in energy consumption. The significance of the factor *non-designated mode* and its positive correlation with the energy consumption metrics imply that improper use of extensions can negatively impact browser performance on energy consumption. Similarly, it is observed that the *fully-inactive mode* significantly jeopardizes the browser performance. **In summary, adhering to proper usage scenarios, i.e., logging in to the extension, granting access to websites, and utilizing extensions on the designated websites, can benefit the overall browser performance.**

Collection of user information by extensions (i.e., adoption of privacy practices) can lead to a significantly negative impact on energy consumption. For example, utilizing *authentication information* within an extension is associated with an increase in the stabilized and page load energy consumption. When *authentication information* is retrieved, it often involves communication with external servers, ongoing data transfers, and periodic page alive detection, leading to increased energy usage. The use of *personal communications* is correlated with increased energy consumption. The *personal communication* practice monitors information, such as emails, texts, and chat messages, which involves continuous background processes and monitoring that may require active network connections and processing power,

resulting in increased energy consumption. Moreover, analyzing and processing personal communications data in real-time can also introduce additional computational overhead, consuming more energy resources. To understand our findings, we post our results to the Chrome extension developer forum, one explains that the additional energy consumption related to *authentication information* and *personal communication* is attributed to the use of APIs and network communications by extensions. Most of the extensions utilize the same thread as that of the browser rather than employing a Worker to perform such tasks. This approach ultimately contributes to higher energy consumption and places an increased load on the browser.

Table 8 The significance of the extension factors for explaining the performance metrics (based on the linear mixed-effects models).

		Page Load Time	Page Load Energy Consumption	Stabilized Energy Consumption
Type	Measurement	Coefficients		
Code Metric	Number of Functions			7.6e-03
	Number of Public Static Methods			1.5e-03
	Code complexity (NPATH - Number of unique paths through a body of code)			11e-03
	Number of Lines of Code		-3.8e-03	-14e-03
	Number of Lines			
	Number of Children/subclasses			-0.47e-03
File Characteristic	Size of .svg files			6.4e-03
	Size of .png files		-4.9e-03	-13e-03
	Size of .eot files		-1.3e-03	-13e-03
	Size of .json files			2.3e-03
	Size of .ttf files			
	Size of .ogg files			3.4e-03
	Size of .map files			
	Size of .jpg files			
	Size of .bak files			-4.5e-03
	Size of .mp3 files			
	Size of .wasm files			2.9e-03
	Size of .scss files			
	Size of .less files			
	Size of .drconf files		1.2e-03	5.3e-03
	Size of .url files			
	Size of .log files			
	Size of .wav files			
Privacy Practice	Location		-0.88e-03	-8.5e-03
	User Activity		4.3e-03	-5.7e-03
	Website Content		3.2e-03	8.5e-03
	Web History		-0.75e-03	
	P.I.I.	-0.20e-03	-2.7e-03	0.88e-03
	Authentication Information	-0.078e-03	4.5e-03	8.7e-03
	Personal Communications		5.1e-03	4.6e-03
User Perspective	# of Users			
	# of Raters			
	Rating Score		0.11e-03	-4.2e-03
Usage Mode	Logged in	-0.32e-03	-1.8e-03	-4.0e-03
	Access to webpage granted		-0.61e-03	-4.2e-03
	Non-designated mode		5.2e-03	6.5e-03
	Fully-inactive mode	0.75e-03	27e-03	

Using *website contents* exhibits a positive correlation in both page load energy consumption and stabilized energy consumption, emphasizing that curtailing the use of *website contents* could result in deteriorated browser performance. Website contents, such as images, scripts, and multimedia elements, can engage users to spend more time on the webpage and require ongoing network requests and data transfers, resulting in additional energy consumption. During a page load, the positive correlation suggests that engaging content may contribute to the increased energy consumption, possibly due to the rendering of dynamic and interactive elements. For stabilized energy consumption, the positive correlation indicates that the ongoing user engagement may involve periodic updates, animations, or dynamic content, which can contribute to sustained energy consumption. However, this may lead to a more satisfying user experience, justifying the positive correlation. Moreover, monitoring *website contents* could lead to increased stabilized energy consumption, as the system remains active to support ongoing interactions and content updates.

While monitoring *location* and *user activity* in the background can be resource-intensive, consuming unnecessary energy, a negative correlation observed between monitoring *location* and *user activity* and the stabilized energy consumption suggests a potential avenue for enhancing browser performance. The collection of *location* and *user activity* information occurs during the page load phase, as evidenced by the positive (i.e., 4.3e-03) and nearly positive (i.e., -0.88e-03) relationships between *location* and *user activity* and the page load energy consumption. Stabilized energy consumption pertains to the period following the page load while the user interacts with the loaded content. By mitigating the energy-intensive background processes associated with location and user activity during the stabilized period, such as reducing the frequency of background processes during low user activity and intensifying it during active user engagement, the browser ensures energy efficiency. Consequently, such actions can significantly improve the stabilized energy consumption, albeit at the cost of a potential negative impact on the page load energy consumption.

A greater number of lines of code (LOC) does not necessarily result in higher energy consumption, but the augmented count of unique paths through the code (NPATH) does. In the Code Metrics group, various metrics are evaluated for their impact on energy consumption. Notably, the number of lines containing source code (LOC) demonstrates a substantial negative effect, as shown in Table 8 (Code Metric), suggesting that an increase in the number of LOC that make up extensions does not necessarily lead to a negative impact on energy consumption (i.e., both stabilized and page load energy consumption) but may exhibit the opposite effect. Plausible explanations are: a larger codebase allows for more efficient and optimized implementations. A well-structured and streamlined codebase, even if extensive, may have undergone optimization practices that enhance energy efficiency. Additionally, a larger codebase indicates a mature and well-maintained extension. Developers who invest time in refining and optimizing their codebase may prioritize user experience as part of their development practices, which could lead to a more resource-efficient extension, resulting in lower stabilized energy consumption.

The positive relationship between code complexity (i.e., NPATH) and the stabilized energy consumption suggests that increased code complexity, as indicated by a higher number of unique paths, is associated with higher stabilized energy consumption. One example of NPATH is shown in Figure 6, and another example is when using the `sscanf` function in C to read data from a string, where can result in n times reading (i.e., executions) through this single statement (i.e., NPATH = n) if the string subsumes n lines. Developers are advised to consider strategies, such as adopting switch statements or hash maps to insert projects to reduce code complexity and steering away from approaches like `sscanf` that may

contribute to elevated stabilized energy consumption. For example, developers can use the `hook` function rather than `sscanf` to read data when needed.

The number of functions is impactful on stabilized energy consumption, indicating a strong positive relationship that an increase in the number of functions in extensions positively correlates with degraded browser performance in terms of stabilized energy consumption. An elevated count of functions in extensions typically leads to heightened overhead execution, including an increase in callbacks and event handling, as well as the introduction of complex interdependencies. This complexity can give rise to frequent interactions and updates, causing the browser to grapple with the management and coordination of various functions. Thus, suboptimal performance during the stabilized phase may ensue that in turn lead to increased energy consumption during the stabilized phase. Additionally, an increase in the number of functions can lead to resource fragmentation. In this context, both the browser and the extension may engage in frequent allocation and deallocation of resources. Due to the unavailability of certain resources, one or more of the available resources remain underutilized or unutilized (Mishra and Bellur, 2016). This resource fragmentation introduces inefficiencies in resource management, thereby adversely affecting energy consumption during the stabilized phase.

Fig. 6 A NPATH example in python.

```
# NPATH = 4: (a,b) — (1,1), (1,0), (0,1), (0,0)
# where (1,1) means functions go_a and go_b are called
# Each if—statement yeilds two paths (i.e., True — call the function and False — do nothing and
    go to the next).
def npathDemo (a, b):
    if a:
        go_a()
    if b:
        go_b()
```

The size of various file types within the source files of extensions exhibit diverse effects on energy consumption. For instance, the file characteristics outlined in Table 8 reveals a negative correlation between the size of .png and .eot typed files and energy consumption, implying that an increase in the utilization of image files (.png) and OpenType font files for webpages (.eot) does not result in heightened energy consumption; instead, it suggests a favorable impact. In contrary to the .png typed files, .svg typed files, which are also image files, have a negative impact on stabilized energy consumption. SVG images consist of a set of instructions that require execution, while png files are composed of pixels that can be loaded more efficiently. It is advisable for developers to take into consideration the impact of file types on browser performance. For instance, it is recommended to substitute the use of .svg files with .png files and consider employing .mp3 or .wav files, which exhibit no significant correlation with browser performance, as opposed to .ogg audio files to enhance the overall browser performance.

Summary: Both the privacy practices and usage modes of extensions significantly impact browser performance. Adhering to proper usage practices, i.e., logging in to the extension, granting access, as well as utilizing the extension on the designated websites, benefits the overall browser performance. Besides, it is crucial for extension developers and users to remain vigilant regarding the impact of privacy practices (i.e., collection of user data) on

energy consumption when adopting or accepting the privacy practices. Extension developers are suggested to consider using separate Worker threads to perform tasks rather than relying on the main browser thread to optimize the extensions. In the context of source code, we recommend that developers focus on reducing code complexity, e.g., adopting hash maps, and minimizing the use of a high number of functions which help mitigate the associated rise in stabilized energy consumption. Furthermore, when constructing extension projects, it is advisable to flexibly employ certain file types, such as PNG and EOT files, to optimize performance.

4 Implications

In this section, we discuss the implications of our findings for extension users, extension and browser developers, and future researchers.

4.1 Implication for Extension Users

Extension users should be aware of the performance ramifications associated with different usage modes of extensions to optimize their configurations of the extensions. As noted in the findings of RQ1 (Section 3.1) and RQ2 (Section 3.2), browser performance is negatively impacted by the use of extensions, irrespective of the specific usage mode of extensions. Such an observation implies that even when extensions are not activated, they can still adversely impact the browser performance. Findings from RQ2 could contribute to establishing best practices for extension users. Extension users should carefully consider the usage mode of extensions while using the extension and only activate the extension that is indispensable. Moreover, they should also periodically scrutinize their extensions and remove any that are no longer needed. By optimizing the configurations of the extensions, users can improve the overall performance and refine the user browsing experience by reducing page load time and minimizing energy consumption.

Extension users could select fungible extensions that have minimal impact on browser performance. As highlighted in the findings of RQ3 (Section 3.3), we discuss the factors that correlate with performance changes. Extension users can select a substitute that offers similar functionality with less impact on the browser performance based on the influential factors (e.g., privacy practices or extension package sizes) as discussed in RQ3. Therefore, users can mitigate the negative impact on browser performance while still fully utilizing the functionality they need.

4.2 Implication for Browser and Extension Developers

Browser and extension developers should pay attention to the performance impact of browser extensions, particularly in regard to energy efficiency. In RQ2 (Section 3.2), we find that using extensions can cause performance deterioration to the browser, which may lead to a poor user experience. Therefore, it is crucial that browser and extension developers prioritize optimizing the performance of their extensions to ensure that their extensions consume minimal resources while still providing the intended functionalities to the users. In addition, developers can gain insights into which usage modes are more likely to influence performance and consider these factors during the development process,

leading to more efficient and reliable extensions. To enhance user satisfaction and decrease the negative impact on the browser performance, we recommend browser and extension developers to take heed of the performance impact of browser extensions by improving the code quality, prioritizing essential content, optimizing resource loading, minimizing data transfers, and utilizing proper privacy practices for the extension. Furthermore, they should regularly maintain their extensions to ensure that extensions are up-to-date and compatible with the latest browser versions.

4.3 Implication for Future Research

Researchers focusing on performance and energy profiling could provide mechanisms to mitigate the adverse performance impact of extensions that are not used in expected scenarios (e.g., when in a non-designated mode), particularly when they are in a fully-inactive mode. As discussed in RQ3 (Section 3.3), highly-rated extensions and extensions with a larger extension size tend to have lower energy consumption. Privacy practices used within extensions can also have varying impacts on the browser performance. Thus, future researchers could verify whether poorly optimized extensions can consume significant amounts of energy and slow down the browser. Such effects can be particularly troublesome for users who have multiple extensions installed, as the cumulative effect of extensions can lead to significant performance degradation. Hence, it is recommended that performance and energy profiling researchers could use our proposed approach in RQ3 (i.e., linear mixed-effects models) to develop mechanisms that can detect when an extension is not being used or not providing useful functionalities to users and automatically deactivate such extension to reduce the performance impact of extensions. Moreover, future work may (1) extend our work to evaluate the performance impact of extensions on other browsers; (2) explore additional factors, such as programming language used and manifest versions, which may influence extension performance; and (3) consider including an additional cluster of extensions that do not disclose privacy practices to further generalize the results and better understand their performance implications.

5 Threats to Validity

In this section, we discuss the threats to the validity of the study.

Threats to Internal Validity. In our study, around 90% of the total extensions, gathered in Section 2.1, are removed. 60% of the total extensions without privacy practice specifications are discarded because of their untraceable structural information for later analysis and noncompliance with Google's policy. Given the vast number of extensions available, it becomes impractical to study each one individually. We opt to cluster the rest of extensions and select the representative one from each category to stand in for the entire group. To ensure diverse representation across the 11 categories in our dataset and to alleviate the impact without adopting privacy practice specifications, we complement 11 additional extensions that developers disclose without adopting any privacy practices. In addition, the sample of extensions studied may not be representative of the broader population of extensions, which could introduce sampling bias. To mitigate this, we keep the selection of extensions diverse, reflecting different categories, popularity levels, the use of different privacy practices, and functionalities. We primarily rely on the median of measurements to assess the performance of browsers. Although more robust against outliers than the mean values, median values may

overlook subtle patterns that the mean might capture but is preferred due to the variability in repeated tests across websites. In RQ3, we use the elastic net regression model to study the relationship between the various factors of an extension and its impact on performance. However, the correlation may not suggest causation. To mitigate this threats, we have proposed 105 possible factors to consider as many factors as possible. Nonetheless, there may still exist other confounding factors not considered in this study.

Threats to External Validity. In this work, we only studied the Google Chrome browser. Our results may not be generalized to other browsers. Nevertheless, as Chrome is the most popular browser, our findings can benefit a large number of browser users and extension developers. In addition, popular browsers such as Chrome and Safari are using similar architectures. Our findings may provide similar insights for other browsers. Moreover, our results may not apply to extensions outside the studied sample although they are highly representative in their use. For example, our results may not generalize to extensions of other browsers or extensions not available on the Chrome Web Store. Compared to prior studies (e.g., 3 extensions tested on Pearce (2020)'s work, 5 extensions tested on Merzdovnik et al. (2017)'s work, and 8 extensions tested on Borgolte and Feamster (2020)'s work), our extension numbers (72) outweigh theirs, and our extensions types are more comprehensive.

Threats to Construct Validity. Threats to the construct validity of our study may involve network instability, running background processes and daemons, such as a routing daemon that handles multiple routing protocols and replaces routed, which may impact the stability of our measurements. It is challenging to achieve complete control over network stability and all background processes or daemons of a system. However, we try to reduce the impact of background processes and daemons as much as possible by stopping them while we collect the measurements. We assume a certain users' usage patterns when we use Selenium to simulate users' interactions with extensions in distinct usage modes of extension. Nevertheless, individual user settings and preferences are unpredictable. Despite our efforts to account for various common conditions and settings, our findings may not be generalizable to cover all the possible real-world user interactions with extensions.

6 Related Work

In this section, we present prior work with respect to browser and extension performance.

6.1 Browser Performance

Studies, such as Macedo et al. (2020), Janssen et al. (2022), and Tian and Ma (2019), investigate the run-time performance and the energy consumption of browsers and web applications. *Macedo et al. (2020)* compare the energy consumption of Google Chrome and Mozilla's Firefox when browsing webpages and find that Google Chrome is more energy-efficient when navigating web pages, but uses more energy for RAM, particularly when interacting with YouTube. *Janssen et al. (2022)* conduct a study to investigate the effect of the Critical CSS technique on the run-time performance and the energy consumption of Android mobile web applications in Google Chrome and Mozilla Firefox. *Janssen et al.* find that the technique can positively improve the run-time performance of Android mobile web apps slightly, but it has no significant impact on energy consumption. *Tian and Ma (2019)* analyze the quality of user experiences across three mobile browsers - Chrome, Firefox, and Opera - by collecting data from 337 webpages and analyzing the loading time and cache performance. *Tian et al.* show that a significant proportion of webpages exhibit notable variations in terms of loading time and cache performance across different browsers.

Prior work (Janssen et al., 2022; Tian and Ma, 2019; Macedo et al., 2020) focuses on examining the browsing performance across various browsers but does not explore the impact of browser extensions on browser performance. Our work, in contrast, studies the impact of extensions on browser performance.

6.2 Extension Performance

In comparison to the studies discussed in 6.1, prior studies, such as Pearce (2020), Merzdovnik et al. (2017), and Borgolte and Feamster (2020), delve deeper into the effect of privacy-focused browser extensions on browser performance. *Pearce (2020)* explores the potential of three open source advertisement (ad) blockers to reduce the page loading time by eliminating ads from internet browsing and video streaming. The evidence indicates that ad blockers are effective in saving energy due to their ability to shorten page loading time. *Merzdovnik et al. (2017)* evaluate the effectiveness and the system performance impact of 5 anti-tracking extensions (e.g., Ad-Block Plus) across 100,000 websites. The results show that these anti-tracking extensions do not increase CPU time, however, they consume more memory. The metrics used in the study, such as memory consumption, are comparable to the performance metrics used in our study. *Borgolte and Feamster (2020)* analyze the impact of eight privacy-focused browser extensions (e.g., Ad-Block Plus and Privacy Badger) on user experience and system performance in both Google Chrome and Mozilla Firefox. Overall, the results indicate that these privacy-conscious extensions do not impede the system performance and even improve the user's browsing experience.

Prior studies (Pearce, 2020; Merzdovnik et al., 2017; Borgolte and Feamster, 2020) predominantly focus on the examination of a single type of extensions, particularly those associated with activity blocking. Our study, however, stands out by encompassing representative extensions from 11 diverse categories, which cover various functional types of extensions. In addition, our study considers a different set of performance metrics (e.g., including energy consumption) and delves deeper to understand how different usage modes and other factors of the extensions (e.g., privacy practices) influence the browser performance. Although our study focuses on a smaller set of websites, the websites used in our experiments are highly representative, as they are selected from 11 categories in Google Chrome web store and commonly utilized by users worldwide. Therefore, our findings still hold the relevance and provide valuable insights into the performance of different extension types across widely-used online platforms.

7 Conclusion

In this paper, we study the impact of extensions on browser performance, specifically in terms of page load time and energy consumption. We observe that browser performance can be negatively impacted by the use of extensions, even when the extensions are used in unexpected circumstances (e.g., not logged in or access to a webpage not granted) or are not active (e.g., not used for designated websites or fully deactivated). We also observe that the privacy practices of an extension are significantly correlated with its performance impact. Our work provides the following recommendations for extension users and extension or browser developers:

– Browser and extension developers should be vigilant about the performance impact of browser extensions, particularly in regard to energy efficiency. For example, they could

provide warnings and information about the performance impact of the extensions or suggest users to disable the extensions in performance-critical scenarios.
- Extension developers and users should be aware of the performance impact of different usage modes of extensions. Users are suggested to adhere to the proper usage practices of extensions (e.g., granting login to an extension when required), while developers are suggested to spend more effort on the performance testing and optimization of the intended usage scenarios of extensions.
- When constructing extension projects for optimal performance, extension developers should be mindful of the influence of code complexity and the usage of certain file types. For example, reducing code complexity (e.g., adopting hash maps) and minimizing the use of a high number of functions are helpful to mitigate the associated rise in stabilized energy consumption. Moreover, it is advisable to flexibly employ certain file types, such as using PNG files in place of SVG, and using EOT files in place of other font files, to optimize performance.
- Extension developers and users should pay attention to the potential performance impact associated with privacy practices when adopting or accepting privacy practices for an extension. Extension developers are advised to optimize extensions by utilizing a dedicated Worker thread for tasks instead of relying solely on the primary browser thread. Extension users, in turn, should be aware that privacy practices (e.g., *website contents* and *personal communication*) could drain more energy consumption, and this consideration should influence their choices when selecting extensions.

In the future, researchers may extend our dataset to include more extensions that do not disclose privacy practices to better understand their performance implications. Furthermore, future work can explore additional factors (e.g., programming language used and manifest versions) that may influence extension performance.

8 Acknowledgment

We would like to thank Dr. Bram Adams and Dr. Stefanos Georgiou for helpful discussions and feedback. The authors acknowledge funding received from the Natural Sciences and Engineering Research Council of Canada (NSERC) and Queen's University.

9 Data Availability

We share our replication package (Jin, 2024) on https://github.com/Bihui-Jin/suppmaterial-impact-of-extensions-on-browser-performance for future work to build on our work.

References

Amazon (2023). *Amazon.com, Inc.* URL: https://www.amazon.co.jp.

Amsel, Nadine and Bill Tomlinson (2010). "Green Tracker: A Tool for Estimating the Energy Consumption of Software". In: *CHI '10 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '10. Atlanta, Georgia, USA: Association for Computing Machinery, pp. 3337–3342. ISBN: 9781605589305. DOI: 10.1145/1753846.1753981. URL: https://doi.org/10.1145/1753846.1753981.

Aniche, Maurício et al. (2016). "SATT: Tailoring Code Metric Thresholds for Different Software Architectures". In: *2016 IEEE 16th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pp. 41–50. DOI: 10.1109/SCAM.2016.19.

Arora, Preeti, Deepali, and Shipra Varshney (2016). "Analysis of K-Means and K-Medoids Algorithm For Big Data". In: *Procedia Computer Science* 78. 1st International Conference on Information Security & Privacy 2015, pp. 507–512. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2016.02.095. URL: https://www.sciencedirect.com/science/article/pii/S1877050916000971.

Banerjee, Nilanjan et al. (2007). "Users and Batteries: Interactions and Adaptive Energy Management in Mobile Systems". In: *UbiComp 2007: Ubiquitous Computing*. Ed. by John Krumm et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 217–234. ISBN: 978-3-540-74853-3.

Benjamini, Yoav and Yosef Hochberg (1995). "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 57.1, pp. 289–300. ISSN: 00359246. URL: http://www.jstor.org/stable/2346101 (visited on 01/10/2023).

Bergstra, James and Y. Bengio (Mar. 2012). "Random Search for Hyper-Parameter Optimization". In: *The Journal of Machine Learning Research* 13, pp. 281–305.

Borgolte, Kevin and Nick Feamster (2020). "Understanding the Performance Costs and Benefits of Privacy-Focused Browser Extensions". In: *Proceedings of The Web Conference 2020*. WWW '20. New York, NY, USA: Association for Computing Machinery, pp. 2275–2286. DOI: 10.1145/3366423.3380292.

Bornholt, James, Todd Mytkowicz, and Kathryn S. McKinley (Aug. 2012). "The model is not enough: Understanding energy consumption in mobile devices". In: *2012 IEEE Hot Chips 24 Symposium (HCS)*, pp. 1–3. DOI: 10.1109/HOTCHIPS.2012.7476509.

Chan-Jong-Chu, Kwame et al. (2020). "Investigating the Correlation between Performance Scores and Energy Consumption of Mobile Web Apps". In: *Proceedings of the Evaluation and Assessment in Software Engineering*. EASE '20. Trondheim, Norway: Association for Computing Machinery, pp. 190–199. ISBN: 9781450377317. DOI: 10.1145/3383219.3383239. URL: https://doi.org/10.1145/3383219.3383239.

Chidamber, S.R. and C.F. Kemerer (1994). "A metrics suite for object oriented design". In: *IEEE Transactions on Software Engineering* 20.6, pp. 476–493. DOI: 10.1109/32.295895.

Cliff, Norman (1993). "Dominance statistics: Ordinal analyses to answer ordinal questions." In: *Psychological Bulletin* 114, pp. 494–509.

Curran-Everett, Douglas (Oct. 2008). "Explorations in statistics: Standard deviations and standard errors". In: *Advances in physiology education* 32, pp. 203–8. DOI: 10.1152/advan.90123.2008.

David, H. et al. (Aug. 2010). "RAPL: Memory power estimation and capping". In: *2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*, pp. 189–194. DOI: 10.1145/1840845.1840883.

David, Howard et al. (2010). "RAPL: Memory Power Estimation and Capping". In: *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design*. ISLPED '10. Austin, Texas, USA: Association for Computing Machinery, pp. 189–194. ISBN: 9781450301466. DOI: 10.1145/1840845.1840883. URL: https://doi.org/10.1145/1840845.1840883.

Desrochers, Spencer, Chad Paradis, and Vincent M. Weaver (2016). "A Validation of DRAM RAPL Power Measurements". In: *Proceedings of the Second International Symposium on Memory Systems*. MEMSYS '16. Alexandria, VA, USA: Association for Computing Machinery, pp. 455–470. ISBN: 9781450343053. DOI: 10.1145/2989081.2989088. URL: https://doi.org/10.1145/2989081.2989088.

Dsouza, Swathi, Jevita Deena Dsouza, and Vanitha T (2017). "Analysis of data using k-means and k-medoids algorithms". In: *International Journal of Latest Trends in Engineering and Technology - Special Issue - SACAIM*, pp. 370–373. ISSN: 2278-621X. URL: https://www.ijltet.org/journal/151065795883.pdf.

Efron, Bradley (1983). "Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation". In: *Journal of the American Statistical Association* 78.382, pp. 316–331. DOI: 10.1080/01621459.1983.10477973. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1983.10477973.

Efron, Bradley and Robert J Tibshirani (1994). *An introduction to the bootstrap*. CRC press.

Fei, Yunsi et al. (2004). "Energy-optimizing source code transformations for OS-driven embedded software". In: *17th International Conference on VLSI Design. Proceedings.* Pp. 261–266. DOI: 10.1109/ICVD.2004.1260934.

Friedman, Jerome, Robert Tibshirani, and Trevor Hastie (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent". In: *Journal of Statistical Software* 33.1, pp. 1–22. DOI: 10.18637/jss.v033.i01.

Georgiou, Stefanos et al. (2022). "Green AI: Do Deep Learning Frameworks Have Different Costs?" In: *Proceedings of the 44th International Conference on Software Engineering*. ICSE '22. Pittsburgh, Pennsylvania: Association for Computing Machinery, pp. 1082–1094. ISBN: 9781450392211. DOI: 10.1145/3510003.3510221. URL: https://doi.org/10.1145/3510003.3510221.

Giardino, Michael and Bonnie Ferri (2016). "Correlating Hardware Performance Events to CPU and DRAM Power Consumption". In: *2016 IEEE International Conference on Networking, Architecture and Storage (NAS)*, pp. 1–2. DOI: 10.1109/NAS.2016.7549395.

Giglio, Cannon and Steven D. Brown (2018). "Using elastic net regression to perform spectrally relevant variable selection". In: *Journal of Chemometrics* 32.8. e3034 CEM-17-0239.R1, e3034. DOI: https://doi.org/10.1002/cem.3034. eprint: https://analyticalsciencejournals.onlinelibrary.wiley.com/doi/pdf/10.1002/cem.3034. URL: https://analyticalsciencejournals.onlinelibrary.wiley.com/doi/abs/10.1002/cem.3034.

Google (n.d.). *Upload videos longer than 15 minutes*. URL: https://support.google.com/youtube/answer/71673?hl=en&co=GENIE.Platform%3DDesktop&oco=0.

Hackeling, Gavin (2017). *Mastering Machine Learning with scikit-learn*. Packt Publishing Ltd.

Hindle, A. (Apr. 2013). "Green mining: a methodology of relating software change and configuration to power consumption". In: *Empirical Software Engineering* 20. DOI: 10.1007/s10664-013-9276-6.

Ihara, Takuya et al. (2015). "Refining Mobile Web Design for Reducing Energy Consumption of Mobile Terminals". In: *2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies*, pp. 13–18. DOI: 10.1109/NGMAST.2015.44.

Jain, Raj (Apr. 1991). *The Art of Computer Systems Performance Analysis: Techniques For Experimental Design, Measurement, Simulation, and Modeling*. en. 1st ed. Nashville, TN: John Wiley & Sons, pp. 216–217. ISBN: 0471503363.

Janssen, Kalle et al. (2022). "On the Impact of the Critical CSS Technique on the Performance and Energy Consumption of Mobile Browsers". In: *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022*. EASE '22. Gothenburg, Sweden: Association for Computing Machinery, pp. 130–139. ISBN: 9781450396134. DOI: 10.1145/3530019.3530033. URL: https://doi.org/10.1145/3530019.3530033.

Jin, Bihui (2024). *Replication Package*. URL: https://github.com/Bihui-Jin/suppmaterial-impact-of-extensions-on-browser-performance (visited on 03/12/2024).

Jin, Bihui, Heng Li, and Ying Zou (2024). *How do Practitioners Perceive Energy Consumption on Stack Overflow?* arXiv: 2409.19222 [cs.SE]. URL: https://arxiv.org/abs/2409.19222.

Kavanagh, Richard and Karim Djemame (2019). "Rapid and accurate energy models through calibration with IPMI and RAPL". In: *Concurrency and Computation: Practice and Experience* 31.13. e5124 cpe.5124, e5124. DOI: 10.1002/cpe.5124. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.5124. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5124.

Khan, Kashif Nizam et al. (Mar. 2018a). "RAPL in Action: Experiences in Using RAPL for Power Measurements". In: *ACM Trans. Model. Perform. Eval. Comput. Syst.* 3.2. ISSN: 2376-3639. DOI: 10.1145/3177754. URL: https://doi.org/10.1145/3177754.

— (Mar. 2018b). "RAPL in Action: Experiences in Using RAPL for Power Measurements". In: *ACM Trans. Model. Perform. Eval. Comput. Syst.* 3.2. ISSN: 2376-3639. DOI: 10.1145/3177754. URL: https://doi.org/10.1145/3177754.

Kor, Ah-Lian et al. (2015). "Applications, energy consumption, and measurement". In: *2015 International Conference on Information and Digital Technologies*, pp. 161–171. DOI: 10.1109/DT.2015.7222967.

Kuhn, Max (2008). "Building Predictive Models in R Using the caret Package". In: *Journal of Statistical Software* 28.5, pp. 1–26. DOI: 10.18637/jss.v028.i05. URL: https://www.jstatsoft.org/index.php/jss/article/view/v028i05.

— (2014). *Futility Analysis in the Cross-Validation of Machine Learning Models*. arXiv: 1405.6974 [stat.ML].

Lin, Kuan-Cheng et al. (2015). "Feature Selection and Parameter Optimization of Support Vector Machines Based on Modified Cat Swarm Optimization". In: *International Journal of Distributed Sensor Networks* 11.7, p. 365869. DOI: 10.1155/2015/365869. eprint: https://doi.org/10.1155/2015/365869. URL: https://doi.org/10.1155/2015/365869.

Lorenz, Mark and Jeff Kidd (1995). "Object-Oriented Software Metrics". In: *SIGSOFT Softw. Eng. Notes* 20.1. Ed. by Will Tracz, pp. 91–93. ISSN: 0163-5948. DOI: 10.1145/225907.773556. URL: https://dl.acm.org/doi/pdf/10.1145/225907.773556.

Macedo, João de et al. (2020). "Energy Wars - Chrome vs. Firefox: Which browser is more energy efficient?" In: *2020 35th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, pp. 159–165. DOI: 10.1145/3417113.3423000.

— (2021). "Energy Wars - Chrome vs. Firefox: Which Browser is More Energy Efficient?" In: *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. ASE '20. New York, NY, USA: Association for Computing Machinery, pp. 159–165.

Mann, Henry B and Donald R Whitney (1947). "On a test of whether one of two random variables is stochastically larger than the other". In: *The annals of mathematical statistics* 18, pp. 50–60.

Manner, Jukka (Dec. 2022). "Black software — the energy unsustainability of software systems in the 21st century". In: *Oxford Open Energy* 2, oiac011. ISSN: 2752-5082. DOI: 10.1093/ooenergy/oiac011. eprint: https://academic.oup.com/ooenergy/article-pdf/doi/10.1093/ooenergy/oiac011/56812916/oiac011.pdf. URL: https://doi.org/10.1093/ooenergy/oiac011.

Merzdovnik, Georg et al. (2017). "Block Me If You Can: A Large-Scale Study of Tracker-Blocking Tools". In: *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 319–333. DOI: 10.1109/EuroSP.2017.26.

Mishra, Mayank and Umesh Bellur (2016). "De-fragmenting the cloud". In: *Proceedings of the 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*. CCGRID '16. Cartagena, Columbia: IEEE Press, pp. 511–520. ISBN: 9781509024520. DOI: 10.1109/CCGrid.2016.21. URL: https://doi.org/10.1109/CCGrid.2016.21.

Moura, I. et al. (May 2015). "Mining Energy-Aware Commits". In: *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, pp. 56–67. DOI: 10.1109/MSR.2015.13.

Murugesan, San (Feb. 2008). "Harnessing Green IT: Principles and Practices". In: *IT Professional* 10, pp. 24–33. DOI: 10.1109/MITP.2008.10.

Palomba, Fabio et al. (Sept. 2018). "On the Impact of Code Smells on the Energy Consumption of Mobile Applications". In: *Information and Software Technology* 105. DOI: 10.1016/j.infsof.2018.08.004.

Pang, C. et al. (May 2016). "What Do Programmers Know about Software Energy Consumption?" In: *IEEE Software* 33.03, pp. 83–89. ISSN: 1937-4194. DOI: 10.1109/MS.2015.83.

Paniego, Juan Manuel et al. (2018). "Analysis of RAPL Energy Prediction Accuracy in a Matrix Multiplication Application on Shared Memory". In: *Computer Science – CACIC 2017*. Ed. by Armando Eduardo De Giusti. Cham: Springer International Publishing, pp. 37–46. ISBN: 978-3-319-75214-3.

Pearce, Joshua M. (2020). "Energy Conservation with Open Source Ad Blockers". In: *Technologies* 8.2. ISSN: 2227-7080. DOI: 10.3390/technologies8020018. URL: https://www.mdpi.com/2227-7080/8/2/18.

Pereira, Rui et al. (2016). "The Influence of the Java Collection Framework on Overall Energy Consumption". In: *Proceedings of the 5th International Workshop on Green and Sustainable Software*. GREENS '16. New York, NY, USA: ACM, pp. 15–21. ISBN: 978-1-4503-4161-5. DOI: 10.1145/2896967.2896968. URL: http://doi.acm.org/10.1145/2896967.2896968 (visited on 09/30/2016).

Pourghassemi, Behnam, Ardalan Amiri Sani, and Aparna Chandramowlishwaran (2019). "What-If Analysis of Page Load Time in Web Browsers Using Causal Profiling". In: *Proc. ACM Meas. Anal. Comput. Syst.* 3.2.

Romano, Daniele et al. (2012). "Analyzing the Impact of Antipatterns on Change-Proneness Using Fine-Grained Source Code Changes". In: *2012 19th Working Conference on Reverse Engineering*, pp. 437–446. DOI: 10.1109/WCRE.2012.53.

Romano, J. et al. (Feb. 2006). "Appropriate Statistics for Ordinal Level Data: Should We Really Be Using t-test and Cohen's d for Evaluating Group Differences on the NSSE and other Surveys?" In: *Annual Meeting of the Florida Association of Institutional Research*, pp. 1–33.

Rousseeuw, Peter (Nov. 1987). "Rousseeuw, P.J.: Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. Comput. Appl. Math. 20, 53-65". In: *Journal of Computational and Applied Mathematics* 20, pp. 53–65. DOI: 10.1016/0377-0427(87)90125-7.

Rousseeuw, Peter J. (1987). "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20, pp. 53–65. ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7. URL: https://www.sciencedirect.com/science/article/pii/0377042787901257.

Schober, Patrick, Christa Boer, and Lothar Schwarte (Feb. 2018). "Correlation Coefficients: Appropriate Use and Interpretation". In: *Anesthesia & Analgesia* 126, p. 1. DOI: 10.1213/ANE.0000000000002864.

Schubert, Erich and Peter J. Rousseeuw (2019). "Faster k-Medoids Clustering: Improving the PAM, CLARA, and CLARANS Algorithms". In: *Similarity Search and Applications*. Springer International Publishing, pp. 171–187. DOI: 10.1007/978-3-030-32047-8_16. URL: https://doi.org/10.1007%5C%2F978-3-030-32047-8_16.

— (2021). "Fast and eager k-medoids clustering: O(k) runtime improvement of the PAM, CLARA, and CLARANS algorithms". In: *Information Systems* 101, p. 101804. ISSN: 0306-4379. DOI: https://doi.org/10.1016/j.is.2021.101804. URL: https://www.sciencedirect.com/science/article/pii/S0306437921000557.

Semrush (2023). *Semrush blog*. URL: https://www.semrush.com/blog/most-visited-websites.

Simon, Noah et al. (2011). "Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent". In: *Journal of Statistical Software* 39.5, pp. 1–13. DOI: 10.18637/jss.v039.i05.

Tay, J. Kenneth, Balasubramanian Narasimhan, and Trevor Hastie (2023). "Elastic Net Regularization Paths for All Generalized Linear Models". In: *Journal of Statistical Software* 106.1, pp. 1–31. DOI: 10.18637/jss.v106.i01.

Thiagarajan, Narendran et al. (2012). "Who killed my battery? analyzing mobile browser energy consumption". In: *Proceedings of the 21st International Conference on World Wide Web*. WWW '12. Lyon, France: Association for Computing Machinery, pp. 41–50. ISBN: 9781450312295. DOI: 10.1145/2187836.2187843. URL: https://doi.org/10.1145/2187836.2187843.

Thorndike, Robert L. (1953). "Who belongs in the family?" In: *Psychometrika* 18, pp. 267–276.

Tian, Deyu and Yun Ma (2019). "Understanding Quality of Experiences on Different Mobile Browsers". In: *Proceedings of the 11th Asia-Pacific Symposium on Internetware*. Internetware '19. Fukuoka, Japan: Association for Computing Machinery. ISBN: 9781450377010. DOI: 10.1145/3361242.3361249. URL: https://doi.org/10.1145/3361242.3361249.

Tiwari, V. et al. (1996). "Instruction level power analysis and optimization of software". In: *Proceedings of 9th International Conference on VLSI Design*, pp. 326–328. DOI: 10.1109/ICVD.1996.489624.

Tong, Haonan, Bin Liu, and Shihai Wang (2018). "Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning". In: *Information and Software Technology* 96, pp. 94–111. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2017.11.008. URL: https://www.sciencedirect.com/science/article/pii/S0950584917300113.

Wang, Peng et al. (July 2022). "Trends in energy consumption under the multi-stage development of ICT: Evidence in China from 2001 to 2030". In: *Energy Reports* 8. DOI: 10.1016/j.egyr.2022.07.003.

Xue, Bing et al. (2016). "A Survey on Evolutionary Computation Approaches to Feature Selection". In: *IEEE Transactions on Evolutionary Computation* 20.4, pp. 606–626. DOI: 10.1109/TEVC.2015.2504420.

Zou, Hui and Trevor Hastie (Mar. 2005). "Regularization and Variable Selection Via the Elastic Net". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 67.2, pp. 301–320. ISSN: 1369-7412. DOI: 10.1111/j.1467-9868.2005.00503.x. eprint: https://academic.oup.com/jrsssb/article-pdf/67/2/301/49795094/jrsssb_67_2_301.pdf. URL: https://doi.org/10.1111/j.1467-9868.2005.00503.x.

A Appendix A: The number of downloads of selected 71 extensions

Table A.1: Download details for 72 selected extensions.

	Extension Name	Extension ID	Number of downloads
1	LM Note Generator For ESPN Fantasy Football	ahcblhpcealjpkmndgmkdnebbjakicno	1,000+
2	Picwatermark	aiiimepjikpdipbpmknolbnjbeohbmaa	22
3	Dark Mode - Night Eye	alncdjedloppbablonallfbkeiknmkdi	200,000+
4	7TV	ammjkodgmmoknidbanneddgankgfejfh	900,000+
5	Neeva Search + Protect for Chrome	aookogakccicaoigoofnnmeclkignpdk	20,000+
6	Ultimate Video Translator	bboamecjefgpaemgfpcjeediamdnkklc	10,000+
7	Better Tab: Speed Dial, News Feed & To-do	behkgahlidmeemjefcbgieigiejiglpc	162
8	Send to Google Maps	bhggankplfegmjjngfmhfajedmiikolo	30,000+
9	Octotree - GitHub code tree	bkhaagjahfmjljalopjnoealnfndnagc	400,000+
10	Truffle.TV (formerly known as Mogul.TV)	bkkjeefjfjcfdfifddmkdmcpmaakmelp	100,000+
11	Tab Resize - split screen layouts	bkpenclhmiealbebdopglffmfdiilejc	700,000+
12	LINER - Search Faster & Highlight Web/Youtube	bmhcbmnbenmcecpmpepghooflbehcack	400,000+
13	Watch2Gether	cimpffimgeipdhnhjohpbehjkcdpjolg	900,000+
14	Adblock for Youtube™	cmedhionkhpnakcndndgjdbohmhepckk	10,000,000+
15	Image Downloader	cnpniohnfphhjihaiiggeabnkjhpaldj	1,000,000+
16	BuiltWith Technology Profiler	dapjbgnjinbpoindlpdmhochffioedbn	300,000+
17	Sourcegraph	dgjhfomjieaadpoljlnidmbgkdffpack	100,000+
18	AmazingHiring	didkfdopbffjkpolefhpcjkohcpalicd	20,000+
19	Ecosia - The search engine that plants trees	eedlgdlajadkbbjoobobefphmfkcchfk	2,000,000+
20	Dark Reader	eimadpbcbfnmbkopoojfekhnkhdbieeh	4,000,000+
21	FrankerFaceZ	fadndhdgpmmaapbmfcknlfgcflmmmieb	1,000,000+
22	Microsoft Rewards	fbgcedjacmlbgleddnoacbnijgmiolem	2,000,000+
23	GoFullPage - Full Page Screen Capture	fdpohaocaechififmbbbbbknoalclacl	5,000,000+
24	Trusted Shops extension for Google Chrome	felcpnemckonbbmnoakbjgjkgokkbaeo	300,000+
25	webpage cloner	ffjnfifmelbmglnajefiipdeejghkkjg	7
26	Tumblr – Post to Tumblr	ffnhmkgpdmkajhomnckhabkfeakhcamm	4,000+
27	Elfster's Elf It!	fhjanlpjlfhhbhbnjohflphmfccbhmoi	10,000+
28	Dynatrace Real User Monitoring	fklgmciohehgadlafhljjhgdojfjihhk	400,000+
29	Use Immersive Reader on Websites	fmidkjgknpkbmninbmklhcgaalfalbdh	100,000+
30	Tricky Enough	fnhmjceoafkkibpijbfpfajbhkknadmb	0
31	FantasyPros: Win your Fantasy League	gfbepnlhpkbgbkcebjnfhgjckibfdfkc	200,000+
32	SimplyCodes | Coupons that work.	gfkpklgmocbcbdabfellcnikamdaeajd	10,000+
33	NekoCap	gmopgnhbhiniibbiilmbjilcmgaocokj	1,000+
34	Pinterest Save button	gpdjojdkbbmdfjfahjcgigfpmkopogic	7,000,000+
35	Wappalyzer - Technology profiler	gppongmhjkpfnbhagpmjfkannfbllamg	1,000,000+
36	coffeelings	hcbddpppkcnfjifbcfnhmelpemdoepkk	200,000+
37	Stem Player Album Upload	iedjpcecgmldlnkbojiocmdaedhepbpn	181
38	Boxel Rebound	iginnfkhmmfhlkagcmpgofnjhanpmklb	1,000,000+
39	Ampie	ikdgincnppajmpmnhfheflannaiapmlm	414
40	Weather	iolcbmjhmpdheggkocibajddahbeiglb	100,000+
41	Enablement Assistant	jbebkmmlkhioeagiekpopmeecaepaihd	400,000+
42	OkTools	jicldjademmddamblmdllfneeaeeclik	100,000+
43	Designer Tools	jiiidpmjdakhbgkbdchmhmnfbdebfnhp	30,000+
44	ER-help Extension	jpefkkpmalfnilnbghfnjodceifpemdb	3,000+
45	Automation 360	kammdlphdfejlopponbapgpbgakimokm	100,000+
46	RSS Reader Extension (by Inoreader)	kfimphpokifbjgmjflanmfeppcjimgah	40,000+
47	Inforness	kgaebnfbgpcnglnhjhglinfiecgccfij	5
48	NFL Live Scores	kimjfkgkpmafgngclkdpjdlkdlghoikh	1,000+
49	Naver/Daum Media Filter(네이버/다음 뉴스 언론사 표시/차단)	kpghljlpdknmomchobaoecdlkcpocaga	7,000+
50	Bulk Image Downloader	lamfengpphafgjdgacmmnpakdphmjlji	30,000+
51	imgur Uploader	lcpkicdemehhmkjolekhlglljnkggfcf	10,000+
52	JobsAlert.pk	ldjnabbinoccbodkejkdiolmadimbjkj	28
53	Lichess Opponent Form	lipplpkgbnhdfdchoibgafjdblpjdkpi	36

Continued on next page

Table A.1 – continued from previous page

	Extension Name	Extension ID	Number of downloads
54	A dónde Viajar	lnphplhkejidgcncalbkbngbiafmjnml	32
55	RotoGrinders - DraftKings Tools	lokmacldfjfgajcebibmmfohacnikhhd	10,000+
56	Feedbro	mefgmmbdailogpfhfblcnnjfmnpnmdfa	40,000+
57	TubeBuddy	mhkhmbddkmdggbhaaaodilponhnccicb	1,000,000+
58	Aerobi - Enhance Your YouTube Workouts	mlfkmhibffpoleieiomjkekmjipdekhg	16
59	Screencastify - Screen Video Recorder	mmeijimgabbpbgpdklnllpncmdofkcpn	6,000,000+
60	WAM: WordSeeker	mpejojclnbakefnlfmnkaaianojbicdk	158
61	CiteMaker CiteWeb | APA 7th Edn.	naankklphfojljboaokgfbheobbgenka	4,000+
62	Keepa - Amazon Price Tracker	neebplgakaahbhdphmkckjjcegoiijjo	2,000,000+
63	MetaMask	nkbihfbeogaeaoehlefnkodbefgpgknn	10,000,000+
64	Bitwarden - Free Password Manager	nngceckbapebfimnlniiiahkandclblb	2,000,000+
65	ShadowPay Trademanager	obhadkdgdffnnbdfpigjklinjhbkinfh	70,000+
66	Custom Cursor for Chrome™	ogdlpmhglpejoiomcodnpjnfgcpmgale	5,000,000+
67	Amazon Assistant for Chrome	pbjikboenpfhbbejgkoklgkhjpfogcam	8,000,000+
68	InteractiveFics	pcpjpdomcbnlkbghmchnjgeejpdlonli	100,000+
69	The Newsroom Beta	pgfokhpgehbmeifbpdhegfnpaahabfja	199
70	買い物ポケット	pgmbeccjfkdbpdjfoldaahpfamjjafma	500,000+
71	Global Twitch Emotes	pgniedifoejifjkndekolimjeclnokkb	100,000+
72	Free Best VPN PC-Chrome-Unlimited Proxy Guide	pkihbahhbihfoebgdfkibnblbhjfgefc	207

Extensions can be visited at the Chrome Web Store via the website https://chrome.google.com/webstore/detail/{ExtensionID}