

Article

Effective Techniques for Protecting the Privacy of Web Users

Maryam Bubukayr ^{1,*} and Mounir Frikha ²¹ College of Computer Science and Information Technology, King Faisal University (KFU), Al-Ahsa 31982, Saudi Arabia² Department of Computer Networks and Communications, King Faisal University (KFU), Al-Ahsa 31982, Saudi Arabia

* Correspondence: 221445328@student.kfu.edu.sa

Abstract: With the rapid growth of web networks, the security and privacy of online users are becoming more compromised. Especially, the use of third-party services to track users' activities and improve website performance. Therefore, it is unavoidable that using personal information to create unique profiles may violate individuals' privacy. Recently, several tools have been developed such as anonymity, anti-tracking, and browser plugins to ensure the protection of users from third-party tracking methods by blocking JavaScript programs and other website components. However, the current state lacks an efficient approach that provides a comprehensive solution. In this paper, we conducted a systematic analysis of the most common privacy protection tools based on their accuracy and performance by evaluating their effectiveness in correctly classifying tracking and functional JavaScript programs, then evaluating the estimated time the browser takes to render the pages for each tool. To achieve this, we automatically browsed the most 50 websites determined in 2022 and categorized them according to different fields to get the in-page (as part of HTML script tags), and all external JavaScript programs. Then we collected data and datasets of 1578 JavaScript elements and obtained six diverse Firefox profiles when the tools were enabled. The results found that Ghostery has the highest percentage of allowing most functioning scripts with the lowest average error rate (AER). While at the same time NoScript achieved the highest percentage of blocking most tracking scripts since it is the highest blocker of third-party services. After that, we examined the speed of the browser finding that, Ghostery improved the load time by 36.2% faster than the baseline, while Privacy Badger only reduced the load time by 7.1%. We believe that our findings can help users decide on a privacy tool that meets their needs. Moreover, researchers and developers can use our findings to improve the privacy of internet users by designing more effective privacy protection techniques.

Keywords: privacy; JavaScript programs; third-party services; security; web tracking

Citation: Bubukayr, M.; Frikha, M. Effective Techniques for Protecting the Privacy of Web Users. *Appl. Sci.* **2023**, *13*, 3191. <https://doi.org/10.3390/app13053191>

Academic Editors: Gianluigi Ferrari and Gianluca Lax

Received: 10 January 2023

Revised: 20 February 2023

Accepted: 27 February 2023

Published: 2 March 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The issue of web tracking has become one of the most significant privacy concerns. It is used by most websites to gather personal information, logins, location, employment status, financial information, gender, medical status, news, or preferences about users to provide high-quality services, sell targeted ads, and even share that data with others without users' consent. The majority of websites contain several web elements that must be retrieved before they can be viewed. When a user opens a browser searching for a certain website, these elements are fetched from the visited site and generate many other HTTP(S) requests for downloading additional elements of the visited site [1].

In this case, web elements can be obtained from two domains, the first-party domain which is the visited site, and third-party domains that might be considered as functional, such as downloading content from a content delivery network (CDN) or serving as tracking domains. Typical examples of web elements are Flash elements, Silverlight, Javascript programs, CSS, java, etc.

Figure 1 demonstrates a general web tracking scenario implemented by JavaScript programs and shows how visitors' data could be silently sent by HTTP headers. When a user searches a URL of example.com and upon retrieving the webpage from a first-party domain, the visitor's browser processes all the HTML tags and every JavaScript program is executed within these tags (steps 1 and 2). When the browser executes JavaScript programs, they enable the browser to send other requests to a variety of websites operated by third-party domains looking for additional content (step 3). The retrieved JavaScript programs can be useful based on the functionalities they provide, such as fetching content from a CDN or it can be tracking such as analytics, ads, etc. Afterward, when the website is completely loaded (step 4), the JavaScript programs can keep track of the visitor's behavior on the website, access the cookies database, or even reconstruct the visitor ID (steps 5 and 6). Finally, JavaScript programs can communicate back to the third-party websites and pass the tracking data to the third-party servers (step 7).

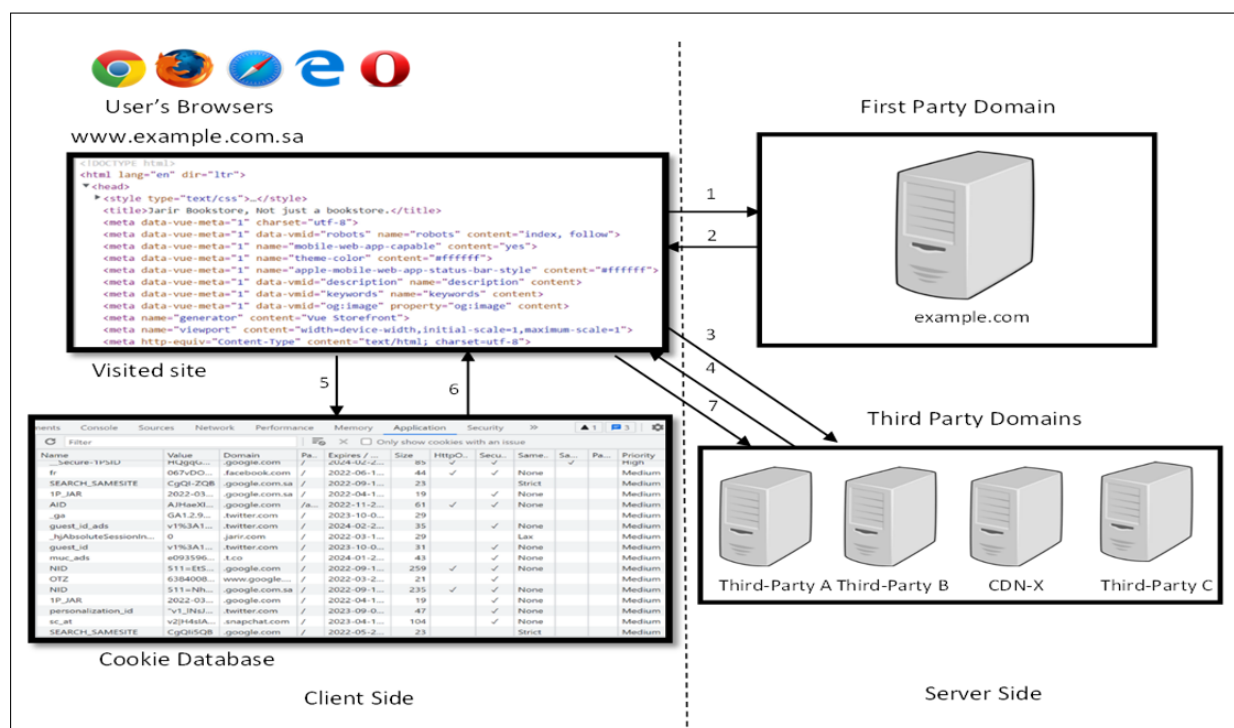


Figure 1. The process of tracking the web page and rendering it through the execution of JavaScript programs. Examples.com uses third-party servers (A, B, CDN-X, and C) looking for additional content and analytics services.

In consequence, JavaScript programs are typically used for various purposes, including data analytics, tracking services, social media platforms, and content distribution networks (CDNs). The content distribution network, also known as a content delivery network, is a network of servers and data centers that are geographically distributed and designed to enhance the performance and availability of online data transmission by the use of caching and other techniques [2]. Nowadays, CDNs are responsible for a large amount of Internet content since they provide the necessary data for websites to function properly, so they should not be classified as trackers, even if they are hosted by third-party domains.

The main goal of this work is to extract tracking and functional JavaScript programs in order to analyze the accuracy and performance of privacy protection tools in terms of achieving the balance between allowing and blocking JavaScript programs. Furthermore, evaluate the estimated time that the browser takes to render the page. The page load time, or PLT using Chrome DevTools.

2. Related Study

A recent and comprehensive overview of the web-tracking domain and guidelines for future studies by outlining the methodologies and importance of the web-tracking field are discussed in Tatiana et al. [3].

Similarly, a detailed overview in Bubukayr and Frikha [4] by outlining potential privacy issues, providing possible tracking mechanisms that could be exploited, and presenting the available privacy-preserving tools with their strengths and weaknesses.

In recent years, several studies [5–7] have addressed the privacy implications of third-party tracking on the web through an analysis of trackers used on each website. Other studies have discussed the existing tracking strategies used by third-party services in order to create profiles of users and track their online activities. Their findings indicate that trackers are widespread as expected, and the web is dominated by Google, Facebook, and Twitter.

Some research studies [1,8–10] have examined and assessed the performance of existing privacy-protection tools.

Dan and Golan [11] analyzed the Ghostery interface, which serves as a privacy protection tool for blocking third-party tracking on websites. The analysis method includes reviewing the extensions' usage and execution and then performing a heuristic analysis of the extension's interface. The findings indicate that researchers do not have any difficulties using Ghostery since they are familiar with it. However, unfamiliar users are unable to experience the full benefits of this extension. Therefore, as a way of easily mitigating privacy breaches, privacy tools must offer a user-friendly interface.

Likewise, the study by Ikram et al. [1] evaluated the performance of five privacy protection tools by measuring their output based on a set of manually labeled JavaScript programs obtained from different domains. The results suggest that existing PPTs need to be substantially improved in order to achieve an optimal balance between true and false positives.

In addition, Cozza et al. [9] conducted a qualitative and quantitative analysis to assess the effectiveness of privacy protection tools in terms of filtering unwanted content, hardware resource consumption, and mean response time. According to the results, existing PPTs are consuming more RAM (Random Access Memory) as they need to maintain the files with resources in the main memory to be able to block them. Due to this, performance seems to be a critical issue considering the high consumption of system resources. Moreover, all existing PPTs mainly utilize blacklists to remove unwanted content. Thus, they do not effectively block web tracking since blacklists need to be constantly updated and maintained by end users.

Moreover, Garimella et al. [10] study the behavior of ad-blocking extensions on different websites, evaluate the advantages of using these extensions and examine how they increase traffic. The result shows that ad-blocking extensions can prevent displaying ads on websites by blocking third-party services and disabling them from being stored on the user's device. However, blocking all third-party services can cause functionality loss on webpages if third-party images, JavaScript programs, or flash files are not loaded.

Also, Muzamil et al. [8] Analyze the efficiency of various tracker blockers in terms of their impact on web quality of experience (QoE) and protection against third-party tracking by generating traffic automatically.

In agreement with our findings, most of the above works also find that current privacy protection tools do not effectively protect against web tracking, cause performance issues, and are difficult to use by end users. Finally, we note that our study on analysing the effectiveness of PPTs is complementary to prior research on their effect on page load time and determining the suitable balance between allowing and blocking web tracking as in [1,8].

3. Background

This section provides an example of some data collection practices by websites and presents an overview of existing privacy-protection tools.

3.1. Data Collection Practice by Websites

Almost every website can generate millions of data points, which are typically collected at two major interfaces, the front-end, and the back-end [12]. The front-end interface can be viewed in the visitors' browser as the website itself. It contains several scripts, primarily written in JavaScript so that it can collect and transfer the data to one or more servers. While the back-end interface is the server that serves the visited website and manages the data it receives from the front-end interface. It is virtually invisible to site visitors, and most are unaware of it. However, it might also be a potential source of data collection.

In order to clarify different front-end data collection methods, the following are some popular JavaScript program codes that are currently used by most websites.

- `navigator.languages`, which could disclose the nationality and the native language of the visitor.
- `navigator.userAgent`, which reveals whether the visitor is on a desktop or mobile device.
- `screen`, which records information about the visitor's screen settings such as height, width, pixel depth, resolution, default orientation, etc.
- `document.location`, which gives valuable details about the visited webpage including the host, protocol, port, the entire URL of the webpage, the origin, and many other details.
- `document.cookie`, which is one of the most discussed methods for tracking website visitors provides a list of all cookies that the website use to track its users.

Finally, all front-end JavaScript programs will be performed automatically by the websites and the data will be collected for every movement of the visitors including every keyboard press, every mouse click and movement, every scrolling activity, every visited website, and much more [13,14]. All this data will be compressed and stored into a JavaScript array, which will be sent to one or more server locations (might include tracking servers) where it can be analyzed and stored.

This work intends to limit the number of tracking JavaScript programs automatically in the front-end interface in order to achieve a minimum level of sensitive data leakage.

3.2. Privacy Protection Tools (PP-Tools)

As part of the huge number of online advertisements that are intrusive and annoying, the market is responding to the users' disapproval against them. Several studies have shown that 78% of internet users dislike advertising on websites [15,16]. That is because they are concerned about the invasion of privacy through online tracking, which is reflected in the rapid growth of ad-blocking tools (Extensions for Chrome and Add-ons for Firefox). They are also called privacy protection tools (PP-Tools). Throughout the years, these tools have been used for a variety of purposes including hiding advertising content or detecting and preventing web tracking techniques on web pages to ensure users' privacy. Many of the privacy protection tools are controlling the execution of JavaScript programs and have much more access to the system features and browsers as JavaScript is embedded within web pages. The advantage of using these tools is that they provide flexibility for users so that removing or installing them can be done easily. However, still cannot achieve the balance between blocking tracking and accepting functional JavaScript programs.

Here we briefly describe and evaluate the most common privacy protection tools (PP-Tools) considered in our experimental analysis study. Table 1 describes the number of users, filtering rules, and statistics of privacy protection tools.

- **Disconnect(DC):** [17] Is a blacklist-based technique primarily used to block 3rd-party tracking cookies and JavaScript programs that are used on social networks like YouTube, Facebook, Instagram, or Twitter. It also provides a Premium version to protect VPN networks and detect malicious software.
- **Ghostery(GT):** [18] Is a powerful privacy extension or add-on launched in 2009 that can detect and block tracking scripts and cookies from webpages in order to improve privacy and focus on only important content. Moreover, it scans the DOM tree with regular expressions for Advertisements, tracking, and other entities stored in a pre-defined blacklist. As a further feature, the users are given the option to enable tracking

do-mains manually with this tool. Therefore, users can activate or deactivate any third-party services within different categories such as Social Media Widgets, Analytics Services, Ads, and so on.

- **Adblock Plus (ABP):** [19] Is primarily focused on blocking unwanted online content including advertisements, banners, pop-ups, videos, and other forms of advertising that are disruptive and bothersome, as well as preventing malware and tracking. It relies on blacklists with a large number of community-maintained rules. Blacklists can be customized to suit the user's needs, such as EasyList (The most popular one that blocks ads from English sites), Fanboy's list (The second most popular Adblock Plus filter list), FR List (which Blocks ads from French sites), or AR List (which Blocks ads from Arabic sites). It also works by searching HTML rendered pages (DOM trees) with regular expressions and blocks the downloading of web resources that refer to blacklisted advertisements and trackers.
- **uBlock (UB):** [20] It is an open-source browser extension for blocking ads and filtering content. It works with most web browsers including Firefox, Chrome, Chromium, Opera, and some versions of Safari. According to some reports [21], uBlock is considered less memory-consuming than any other extension that offers similar functionality. It uses the same lists of the previous extensions including EasyList, Peter Lowe, Malware Domains, EasyPrivacy, etc. As well as users can set their own filtering preferences.
- **Privacy Badger(PB):** [22] It uses an internal blacklist and a heuristic algorithm to block different types of third-party tracking, including canvas fingerprinting, local storage super cookies, and identifying cookies. In comparison to other PPTs, PB does not require any prior knowledge, configurations, or setup on the user's part. It works by sending the Do Not Track header with every page request, and assessing whether the user is still being tracked. Upon determining that the probability is too high, the algorithm automatically denies the request for a third-party domain.
- **NoScript(NS):** [23] A white list-based tool allows content that has been explicitly authorized by the user, so the default behavior is to block all website content. The rules are expressed as regular expressions, similar to Adblock Plus [19], and It works by blocking all executable content on a webpage that may threaten security including Java programs, Flash, Silverlight, JavaScript programs, and more. However, this may result in usability problems and requires considerable user interaction.

Table 1. Filtering rules and statistics of the Privacy protection tools.

Add-On or Extension	Users Base	Rule-Based Filtering
Disconnect	600,000+ users	Blacklist
Ghostery	2,000,000+ users	Blacklist
Adblock Plus	10,000,000+ users	EasyList
uBlock	700,000+ users	Blacklist
NoScript	100,000+ users	Whitelist
Privacy Badger	1,000,000+ users	Heuristic algorithm

4. Methodology

This section presents the methodology for collecting data and the datasets we obtained for evaluating the privacy-protection tools. Part 1 covers the target websites that were used for achieving the metrics. Part 2 includes a list of all experiment tools and settings that were utilized during data collection and Part 3 describes the implementation of the PPTs measurement.

4.1. Analysis of Websites

In our study, the ranking of the site list was sorted out based on Alexa.com Top 50 Saudi Arabia and global websites determined in 2022 and categorized according to different fields, including Search Engines, News and Social Networks, E-Commerce, Law

and Government, Dictionaries and Encyclopaedias, Online Designs, Banking, Education and Games. Table 2 shows the crawled websites with their categories as well as the total number of websites Alexa found linking to this site and the result indicates that the most popular Saudi Arabia website was google.com.sa with 6,458,120 sites linking in. Figure 2 summarizes the distribution of the aggregators discovered on the most 50 websites during the crawling process and the result shows that facebook.com is the leading family of controlled aggregators with 11,492,297 sites linking in. While qiwa.sa site occupies the least aggregator with 23 sites linking in.

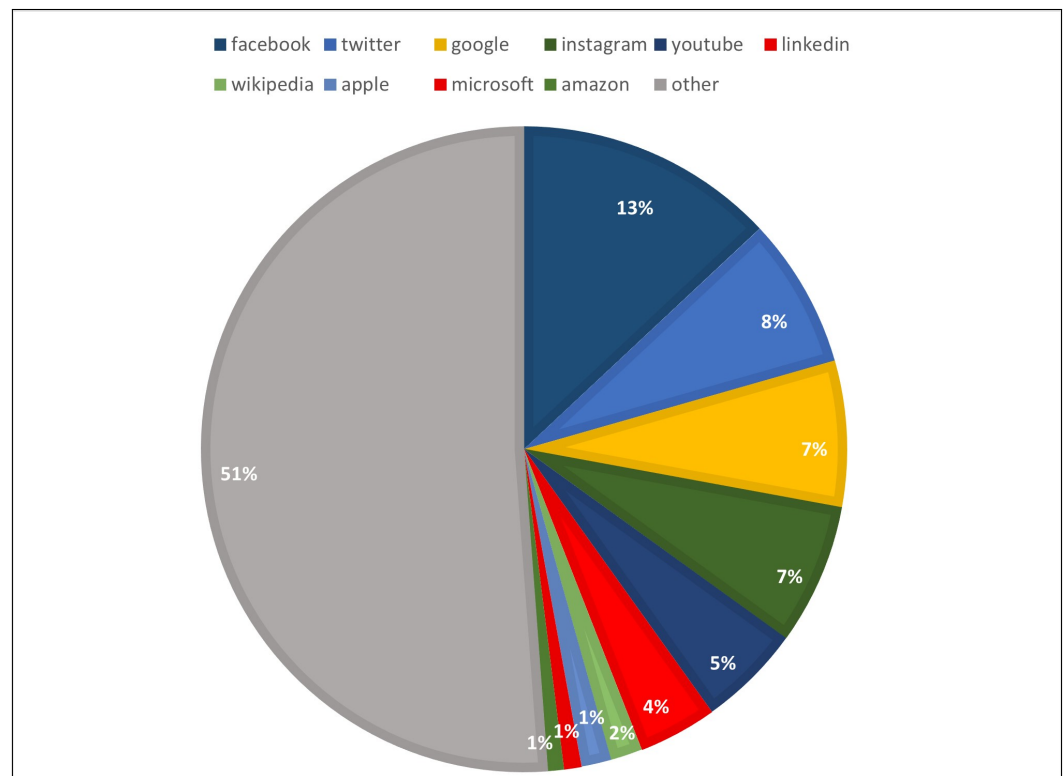


Figure 2. The top 10 aggregators observed during the analysis process.

Table 2. Top 50 websites from different categories visited by SA users (all links were automatically accessed on 29 June 2022).

No.	The Website	Category	Total Sites Linkingin
1	www.google.com	Search Engines	6,458,120
2	www.google.com.sa		6735
3	www.youtube.com	News and Social Networks	4,562,408
4	www.twitter.com		6,666,485
5	www.facebook.com		11,492,297
6	www.whatsapp.com		705,088
7	www.instagram.com		6,238,426
8	www.twitter.com		6,666,485
9	www.linkedin.com		3,467,887
10	www.maktoob.yahoo.com		282,610
11	www.tiktok.com		118,620
12	www.netflix.com		56,631
13	www.argaam.com		1911
14	www.zoom.us		235,425
15	www.snapchat.com		33,546
16	www.alwatan.com.sa		2693
17	www.okaz.com.sa		1778

Table 2. Cont.

No.	The Website	Category	Total Sites Linkingin
18	www.telegram.org	E-commerce	16,015
19	www.mbc.net		644
20	www.haraj.com.sa		342
21	Amazon.sa		359
22	www.ar.aliexpress.com		28,093
23	www.amazon.com		709,590
24	www.microsoft.com		771,705
25	www.almubasher.com.sa		53
26	www.office.com		100,406
27	www.noon.com		553
28	www.microsoftonline.com		13,309
29	www.myshopify.com		104
30	www.salla.sa		286
31	www.apple.com		1,313,163
32	www.github.com		303,583
33	www.zid.sa		33
34	www.jarir.com		279
35	www.ilovepdf.com		1857
36	www.absher.sa	Law and Government	160
37	www.iam.gov.sa		40
38	www.gosi.gov.sa		128
39	www.mc.gov.sa		60
40	www.qiwa.sa		23
41	www.najiz.sa		25
42	www.mol.gov.sa		175
43	www.hrsd.gov.sa		201
44	www.wikipedia.org	Dictionaries and Encyclopedias	1,409,659
45	www.canva.com	Online Designs	43,407
46	www.alinma.com	Banking	74
47	new.alahlionline.com		34
48	www.moe.gov.sa	Education	569
49	www.wajibati.net		826
50	www.twitch.tv	Games	70,337

4.2. Experimental Setup

The test tools that are used for our dataset collection and analysis of the PPTs:

- A set of the most common 50 websites to extract unique JavaScript programs.
- Six of the most common and well-known freeware browser plug-ins (Disconnect, Ghostery, Adblock Plus, uBlock, NoScript, and Privacy Badger).
- Tests were performed on a Linux x64 Ubuntu Operating System and jupyter notebook for running the Python code.
- An open-source web browser with an associated webdriver for automated testing of websites. In our experiment, we used the Firefox browser with geckodriver to test sites automatically.

4.3. Data Collection

For the crawl of the most 50 websites, We simulate the experience of browsing websites on a device using Selenium webdriver [24], a toolset that allows us to retrieve all elements of a DOM tree (a rendered HTML page). Using Python scripts and a connector for the most popular web browsers, we automated Firefox browsing to download websites automatically. During the crawling process, our crawler waits 100 s for each target website to be loaded.

Then, we dump the DOM tree for each webpage looking for JavaScript programs called JSec programs when the privacy protection tools are disabled as a baseline for the remainder of the work, referred to as PPTs Off. Through this crawling process, we were able to build a dataset of 1578 JavaScript elements. At the same time, we create a separate Firefox profile for each tool to simultaneously fetch the JSec programs when the PPTs are enabled. The extensions were installed manually which made a new Firefox profile. As a result, we obtain six diverse Firefox profiles from the official Firefox extension page. Then, we applied the default settings to each except for Adblock Plus and Ghostery. In Adblock Plus, we have included an Arabic language filter list along with EasyList. Also, we have activated the recommended filters (Block additional tracking, Block cookie warnings, Block push notifications, and Block social media icon tracking). In Ghostery, the default setting does not enable any filtering functionality to keep web pages from being tracked. Instead, the user must manually configure the setting to protect against web tracking [11].

All these steps are followed to test our experiment and preserve websites from being tracked. To evaluate the effectiveness of the PPTs in terms of blocking or enabling tracking JSec programs. We performed a quantitative analysis to quantify the number of blocked and enabled tracking JSec programs when handling requests from the most common 50 services. Considering the dynamic nature of the websites and different advertisements appearing at different times when accessing the sites, the experiment was performed simultaneously to ensure that the number of JSec programs remains constant. (Figure 3) summarizes the general structure of the analysis process.

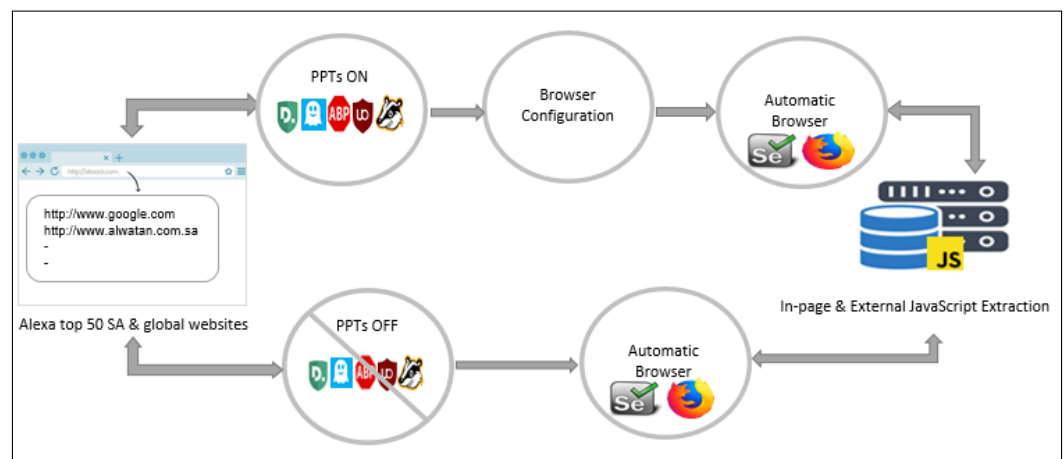


Figure 3. Measurement framework for the privacy protection tools.

5. Experimental Results

In this study, we are interested in evaluating the effectiveness of the privacy protection tools PPTs by measuring the effect on the performance the users perceived and how fast the time it takes to load a web page by blocking third-party requests. Therefore, we extracted JavaScript programs, one of the most important web elements that are typically used for tracking services.

Figure 4 illustrates the total number of JavaScript programs extracted from the top 50 Alexa websites. In the absence of ad-blocking extensions, we obtained 1578 Javascript programs. Furthermore, the result presents the total number of allowed JavaScript programs for NoScript, Privacy Badger, uBlock, Adblock Plus, Ghostery, and Disconnect extensions when they are enabled and configured.

For in-page JavaScript programs, we obtained 828 JSec and we found that disconnect blocked more than 40% of the embedded script tags that are necessary for providing high availability and high performance of web pages, which makes them ineligible for being classified as trackers. Followed by Privacy Badger, Adblock Plus, and uBlock with 36%, 24%, and 17% of blocking the essential scripts respectively. As well, Ghostery and NoScript have the lowest percentage of blocking functioning scripts, each with 11%. For more details, (see Table 3).

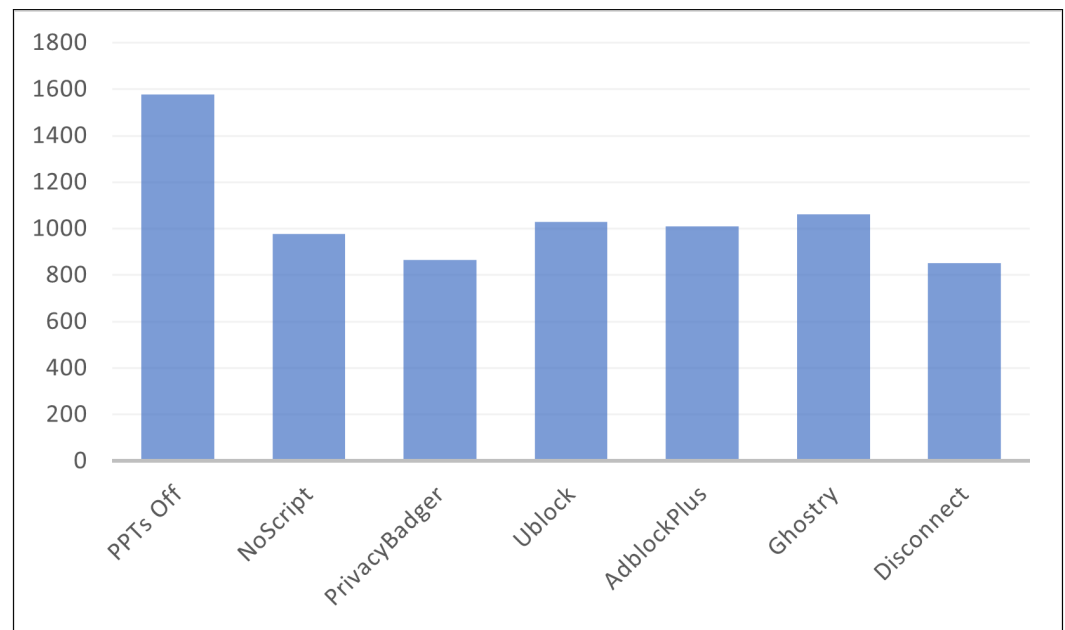


Figure 4. The total number of JavaScript programs extracted from the 50 domains.

Table 3. The percentage of all allowed and blocked in-page JavaScript programs.

JSec	PPTs Off	DC	GT	ABP	UB	PB	NS
In-page	828	467	691	590	646	496	698
Blocked	-	40%	11%	24%	17%	36%	11%
Allowed	-	60%	89%	76%	83%	64%	89%

For external JavaScript programs, we obtained 750 JSec and we found that NoScript is the highest blocker of the external third-party services with 61%. The next two observations involve Privacy Badger and Ghostery with 48% blocking, followed by Disconnect and uBlock with 47% and 44% blocking respectively. And finally, Adblock Plus provides the least amount of blocking third-party services with only 42%. For more details, (see Table 4).

Table 4. The percentage of all allowed and blocked external JavaScript programs.

JSec	PPTs Off	DC	GT	ABP	UB	PB	NS
External	750	385	372	421	403	373	278
Blocked	-	47%	48%	42%	44%	48%	61%
Allowed	-	53%	52%	58%	56%	52%	39%

Further analysis is performed to assess the accuracy of the privacy protection tools by measuring the Sensitivity and Specificity based on our labeled dataset of 50 websites. For this, we measure the true positive (TP), false positive (FP), true negative (TN), and false negative (FN) rates of each of the PPTs as shown in Table 5.

Sensitivity is defined as the probability of correctly identifying the existence of a particular feature or condition. This is also known as the true positive rate (TPR). Consider our case of allow and block functional, Sensitivity relates to the ability of PPTs to efficiently allow functional JavaScript programs. In mathematic terms, this can be expressed as:

$$\text{Sensitivity (TPR)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

where True Positive (TP) is the number of functional JavaScript programs that are correctly identified, False Negative (FN) is the number of tracking JavaScript programs that are incorrectly allowed.

Table 5. Comparison of the PPTs with our labeled dataset between allow and block functional (TP, FP), and allow and block tracking (FN, TN).

PPTs	Positive Class (Functional Jsec)		Negative Class (Tracking Jsec)	
	TPR	FPR	TNR	FNR
DC	0.69	0.31	0.68	0.32
GT	0.87	0.13	0.84	0.16
ABP	0.82	0.18	0.73	0.27
UB	0.86	0.14	0.82	0.18
PB	0.71	0.29	0.69	0.31
NS	0.80	0.20	0.87	0.13

While Specificity is defined as the probability of correctly rejecting the existence of a particular feature or condition. This is also known as the true negative rate (TNR). Consider our case of allow and block tracking, Specificity relates to the ability of PPTs to efficiently block tracking JavaScript programs. In mathematic terms, this can be expressed as:

$$\text{Specificity (TNR)} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

where True Negative (TN) is the number of tracking JavaScript programs that are correctly blocked, False Positive (FP) is the number of functional JavaScript programs that are incorrectly blocked. (Figure 5) illustrate the accuracy of the privacy protection tools based on Sensitivity and Specificity.

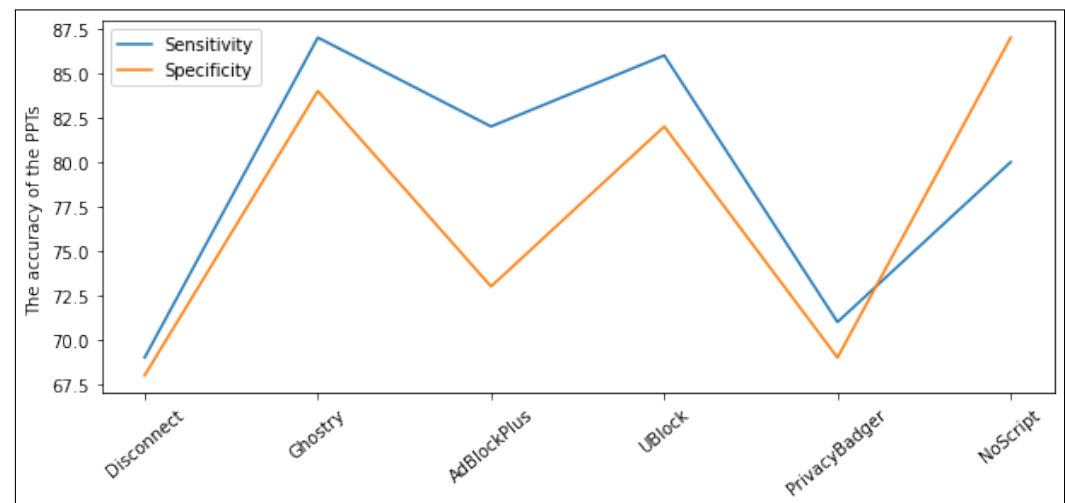


Figure 5. Shows the ratio between sensitivity and specificity of the PPTs.

We considered sensitivity and specificity to help us decide which privacy tool would best identify negative and allow positive JavaScript programs. If correctly identifying positives is more important, then we should choose a privacy tool with higher sensitivity. In contrast, if correctly identifying negatives is more critical, then specificity should be prioritized.

Based on the results, we can conclude that PP-Tools' true positive rates vary from 69% to 87%, and true negative rates vary from 68% to 87%. As expected, Ghostery has the highest Sensitivity rate of 87% due to allowing most JavaScript programs to run in-page, at

the expense of its poorest Specificity rate. The next best are uBlock, Adblock Plus, NoScript, Privacy Badger, and Disconnect with 86%, 82%, 80%, 71%, and 69% respectively. In spite of this, NoScript achieves the highest specificity rate of 87% since it is the highest blocker of external third-party services. Followed by Ghostery, uBlock, Adblock Plus, Privacy Badger, and Disconnect with 84%, 82%, 73%, 69%, and 68% respectively. Furthermore, we observed that Ghostery achieved the lowest average error rate (AER) of 0.145, which is calculated as the average of false positive and false negative rates. However, Disconnect achieved the highest average error rate of 0.315 due to its poorest sensitivity and specificity. The average error rate (AER) reports in (Figure 6).

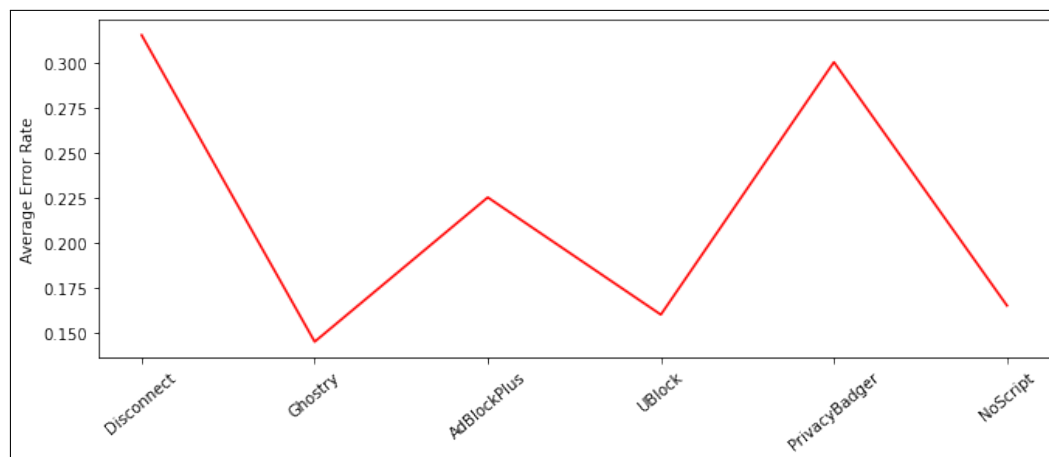


Figure 6. The average error rate (AER) of the PPTs.

The next focus is how fast the browser can render the pages. The page load time or PLT using Chrome DevTools [25], as shown in (Figure 7). Chrome DevTools is a collection of web developer tools integrated directly into the Google Chrome browser. It helps edit pages directly from within the browser and diagnose problems easily. Also, it includes prototyping CSS, debugging CSS and JavaScript, and analyzing load performance. A general conclusion of the study is that privacy protection tools improve the loading time of web pages by blocking incoming requests from third-party services. Comparing initial results (PPTs Off) with each tool, we observe that PPTs decrease the time it takes to render the page by filtering many third parties to contact. Ghostery improves the average load time by 36.2% over the standard browser. Privacy Badger, AdBlock Plus, Disconnect, and uBlock increase the loading speed of the pages by 28.4%, 23.5%, 23.3%, and 22.6% respectively. Furthermore, NoScript also reduces load time with a minor improvement of 7.1%.

Further analysis is that we investigated the presence of obfuscation in JavaScript programs. The purpose is to verify the use of obfuscation in web tracking, and how it affects privacy protection tools' classifications. To the best of our knowledge, this is the first analysis study of the PPTs in terms of their effectiveness in detecting obfuscated JavaScript programs. Obfuscation in JavaScript involves transforming the visual representation of the code into an altered version that is extremely difficult to read, understand, and reverse-engineer without changing its functional characteristics [26]. Web trackers commonly apply obfuscation methods in JavaScript codes to avoid detection by various privacy protection software and conceal their malicious intentions. Web developers may also use obfuscation and code protection methods to protect their code. However, obfuscation can be utilized by malware writers and attackers to bypass security measures. In (Figure 8) we observed that 30% of the tracking JavaScript programs are obfuscated while 70% are plain-text code. We also observed that 36% of the functional JavaScript programs are obfuscated while 64% are plain-text code.

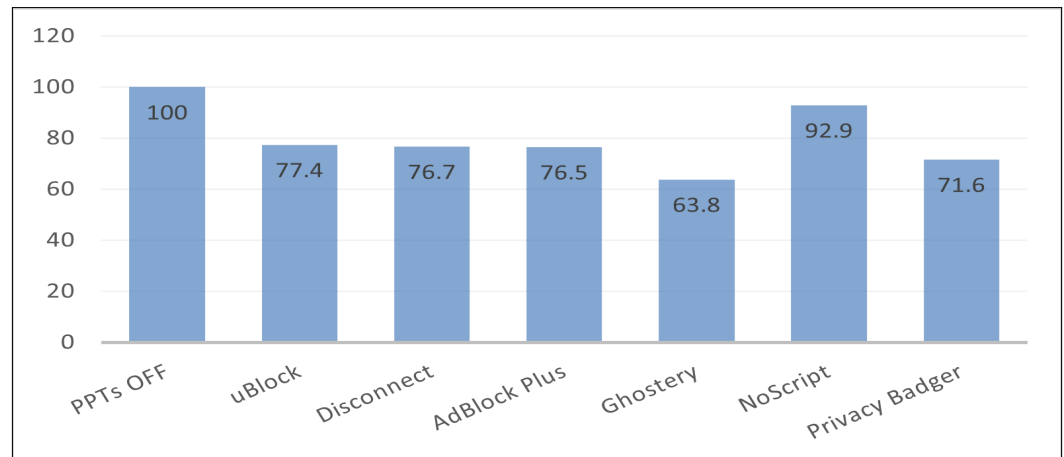


Figure 7. The average page load time, or PLT.

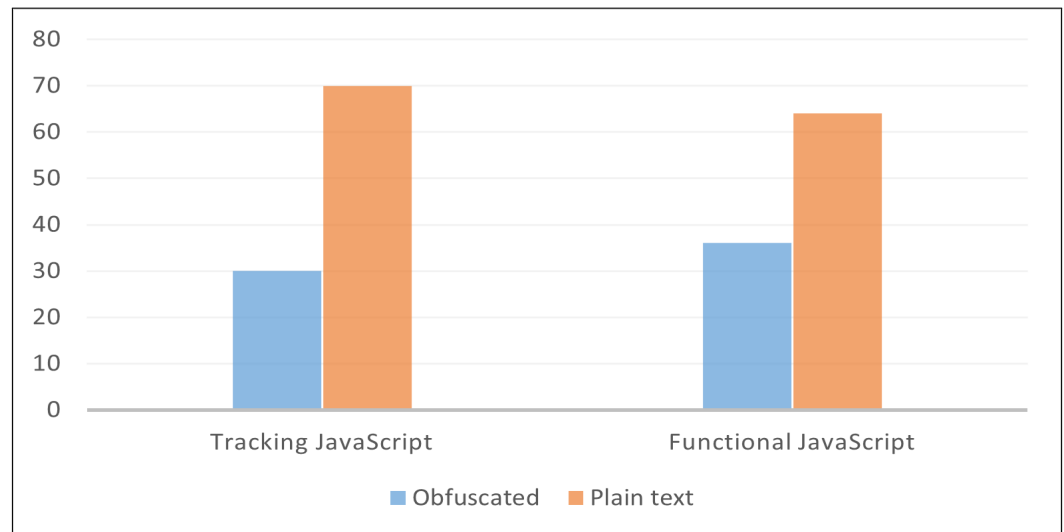


Figure 8. The ratio of obfuscated JavaScript programs presents on the top 50 websites.

6. Discussion

In particular, this study investigates how users of the most common Saudi Arabia websites are tracked, how their privacy is exposed, and how to protect users' privacy according to their needs by evaluating which publicly available privacy protection methods that can reduce tracking with low impact on users' performance. In order to achieve this, we automated browsing the top 50 Saudi Arabia websites according to Alexa.com and categorized them into nine categories. The results of the long-term crawling process identified 1578 unique JavaScript elements on the most visited websites in Saudi Arabia. On average, there were 45 tracking elements per site was found within the top 50 websites. This indicates that the scale of the problem we are addressing is significant. Misclassifying a small portion could negatively affect users' experience. Another scientific contribution is the evaluation of the existing privacy protection methods in order to guide users toward a privacy tool that is suitable for their needs. In order to achieve this, we conducted a systematic analysis of the most common PPTs based on their accuracy, performance, and effectiveness in detecting different forms of obfuscated JavaScript programs by analyzing a new dataset of 1578 JavaScript elements generated by different tools. The results indicated that Ghostery had the highest sensitivity rate of permitting functional scripts with the lowest error rate, as well as better browser speed. While NoScript achieved the highest specificity rate of blocking tracking scripts since it is the highest blocker of third-party services. Further results indicated that the majority of the privacy protection tools are ineffective in detecting JavaScript programs that are obfuscated. The reason behind this is

that all existing privacy solutions heavily rely on the use of a static blacklist to filter out unwanted content. Therefore, when the JavaScript program becomes more complicated with multiple forms of obfuscation it can easily overcome the detection. Recent work has implemented machine learning techniques with blacklisting into PPTs to automatically detect tracking and functional JavaScript programs [1,9]. However, existing approaches have not adequately addressed the obfuscation problem [1,27]. To the best of our knowledge, no previous research has been conducted to detect obfuscated tracking and obfuscated functional JavaScript programs. Finally, as future work, we are planning to characterize JavaScript programs' obfuscation techniques to enhance the resiliency of the PPTs by implementing different machine learning techniques.

7. Conclusions

The use of privacy-protection tools to ensure users' privacy has been widely explored; however, there exists no comprehensive tool that results in a more efficient, user-friendly, and effective solution. In conclusion, the findings indicate that current PPTs are inefficient at maintaining a good balance between preventing tracking and allowing essential functionalities of websites. Several factors may contribute to this weakness. For instance, current PPTs still rely on black lists or pre-defined lists of URL patterns that block JavaScript execution for third-party services. In order for blacklists to be continuously updated and maintained, human effort is required. Further problem is that most PPTs fail to inspect obfuscated JavaScript programs efficiently, and thus allow obfuscated tracking Scripts to bypass successfully.

This paper introduced a systematic comparison of the most popular privacy protection tools in terms of achieving the balance between allowing and blocking functional and tracking JavaScript programs. The results found that Ghostery has the highest percentage of allowing most functioning scripts at 87% with the lowest average error rate (AER) of 0.145. However, Ghostery needs to be configured manually to provide effective protection against third-party services, which is difficult and time-consuming for end-users. While simultaneously, NoScript achieves the highest percentage of blocking most tracking scripts at 87%, since it is the highest blocker of external third-party services.

After that, we examined the speed of the browser to render the web page finding that, blocking third-party services through PPTs substantially reduced the time it takes to load a web page when compared with the original result (PPTs Off). For example, Ghostery improves the average load time by 36.2% faster than the baseline, while Privacy Badger reduces the load time by only 7.1%.

We believe that our study can help researchers and developers gain a comprehensive background about PPTs so they can devise an efficient approach to improve the detection rate and get an optimal balance between allow-functioning and block-tracking JavaScript programs. Furthermore, our findings can guide users in deciding on a privacy tool that is appropriate to their needs.

Author Contributions: Conceptualization, M.B.; Methodology, M.B.; Software, M.B.; Validation, M.B.; Formal analysis, M.B.; Investigation, M.B. and M.F.; Resources, M.B.; Data curation, M.B.; Writing—original draft, M.B.; Writing—review & editing, M.F.; Visualization, M.B.; Supervision, M.F.; Project administration, M.F.; Funding acquisition, M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by King Faisal University, Saudi Arabia [Project No. GRANT2,932].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This work was supported through the Annual Funding track by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia [Project No. GRANT2,932].

Conflicts of Interest: The authors declare no conflict of interest.

References

- Ikram, M.; Asghar, H.J.; Kaafar, M.A.; Krishnamurthy, B.; Mahanti, A. Towards Seamless Tracking-Free Web: Improved Detection of Trackers via One-class Learning. *arXiv* **2017**, arXiv:1603.06289.
- Xu, K.; Li, X.; Bose, S.K.; Shen, G. Joint Replica Server Placement, Content Caching, and Request Load Assignment in Content Delivery Networks. *IEEE Access* **2018**, *6*, 17968–17981. [\[CrossRef\]](#)
- Ermakova, T.; Fabian, B.; Bender, B.; Klimek, K. Web tracking-A Literature Review on the State of Research. In Proceedings of the 51st Hawaii International Conference on System Sciences, HICSS 2018, Hilton Waikoloa Village, HI, USA, 3–6 January 2018.
- Maryam Abdulaziz Saad Bubukayr, M.F. Web Tracking Domain and Possible Privacy Defending Tools: A Literature Review. *J. Cyber Secur.* **2022**, *4*, 79–94. [\[CrossRef\]](#)
- Englehardt, S.; Reisman, D.; Eubank, C.; Zimmerman, P.; Mayer, J.; Narayanan, A.; Felten, E.W. Cookies That Give You Away: The Surveillance Implications of Web Tracking. In Proceedings of the 24th International Conference on World Wide Web (WWW'15), Florence, Italy, 18–22 May 2015; International World Wide Web Conferences Steering Committee: Geneva, Switzerland, 2015; pp. 289–299.
- Kalavri, V.; Blackburn, J.; Varvello, M.; Papagiannaki, K. Like a Pack of Wolves: Community Structure of Web Trackers. In Proceedings of the International Conference on Passive and Active Network Measurement (PAM), Heraklion, Greece, 31 March–1 April 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 42–54.
- Schelter, S.; Kunegis, J. Tracking the Trackers: A Large-Scale Analysis of Embedded Web Trackers. In Proceedings of the 10th International AAAI Conference on Web and Social Media (ICWSM 2016), Cologne, Germany, 17–20 May 2016; pp. 679–682.
- Muzamil, M.; Khan, A.; Hussain, S.; Jhandir, M.Z.; Kazmi, R.; Bajwa, I.S. Analysis of Tracker-Blockers Performance. *Pak. J. Eng. Technol.* **2021**, *4*, 184–190.
- Cozza, F.; Guarino, A.; Isernia, F.; Malandrino, D.; Rapuano, A.; Schiavone, R.; Zaccagnino, R. Hybrid and Lightweight Detection of Third Party Tracking: Design, Implementation, and Evaluation. *Comput. Netw.* **2020**, *167*, 106993. [\[CrossRef\]](#)
- Garimella, K.; Kostakis, O.; Mathioudakis, M. Ad-blocking: A study on performance, privacy and counter-measures. In Proceedings of the ACM on Web Science Conference, WebSci'17, New York, NY, USA, 25–28 June 2017; pp. 259–262.
- Bouhnik, D.; Carmi, G. Interface Application Comprehensive Analysis of Ghostery. *Int. J. Comput. Syst.* **2018**, *5*, 4–10.
- Oulasvirta, A.; De Pascale, S.; Koch, J.; Langerak, T.; Jokinen, J.; Todi, K.; Laine, M.; Kristhombuge, M.; Zhu, Y.; Miniukovich, A.; et al. Aalto Interface Metrics (AIM) A Service and Codebase for Computational GUI Evaluation. In Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings, Berlin, Germany, 14–17 October 2018; pp. 16–19.
- Malandrino, D.; Petta, A.; Scarano, V.; Serra, L.; Spinelli, R.; Krishnamurthy, B. Privacy awareness about information leakage: Who knows what about me? In Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society, WPES13, Berlin, Germany, 4 November 2013; pp. 279–284.
- Wang, Y.; Cai, W.d.; Wei, P.C. A deep learning approach for detecting malicious JavaScript code. *Secur. Commun. Netw.* **2016**, *9*, 1520–1534. [\[CrossRef\]](#)
- Pujol, E.; Hohlfeld, O.; Feldmann, A. Annoyed users: Ads and ad-block usage in the wild. In Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC, Tokyo, Japan, 28–30 October 2015; pp. 93–106.
- Mathur, A.; Vitak, J.; Narayanan, A.; Chetty, M. Characterizing the Use of {Browser-Based} Blocking Extensions To Prevent Online Tracking. In Proceedings of the 14th USENIX Conference on Usable Privacy and Security, Baltimore, MD, USA, 12–14 August 2018; pp. 103–116.
- Disconnect. Available online: <https://disconnect.me> (accessed on 20 March 2022).
- Ghostery. Available online: <https://www.ghostery.com> (accessed on 20 March 2022).
- Adblock Plus | The world's #1 Free Ad Blocker. Available online: <https://adblockplus.org/> (accessed on 20 March 2022).
- uBlock Origin—Free, Open-Source ad Content Blocker. Available online: <https://ublockorigin.com/> (accessed on 20 March 2022).
- uBlock, the Memory-Friendly Ad-Blocker, Is Now Available for Firefox. Available online: <https://lifelacker.com/ublock-the-memory-friendly-ad-blocker-is-now-availabl-1681818949> (accessed on 20 March 2022).
- Privacy Badger. Available online: <https://privacybadger.org/> (accessed on 20 March 2022).
- What is It?—NoScript: Block Scripts and Own Your Browser! Available online: <https://www.noscript.net/> (accessed on 20 March 2022).
- What is Selenium? Available online: <http://www.seleniumhq.org/> (accessed on 20 March 2022).
- Chrome DevTools Protocol.(n.d.). Chrome DevTools Protocol. Available online: <https://chromedevtools.github.io/devtools-protocol/> (accessed on 20 March 2022).
- Alazab, A.; Khraisat, A.; Alazab, M.; Singh, S. Detection of Obfuscated Malicious JavaScript Code. *Future Internet* **2022**, *14*, 217. [\[CrossRef\]](#)
- Masood, R.; Vatsalan, D.; Ikram, M.; Kaafar, M.A. Incognito: A Method for Obfuscating Web Data. In Proceedings of the 2018 World Wide Web Conference, WWW'18, Lyon, France, 23–27 April 2018; International World Wide Web Conferences Steering Committee: Geneva, Switzerland, 2018; pp. 267–276.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.