Received July 1, 2018, accepted August 5, 2018, date of publication September 10, 2018, date of current version October 12, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2869251

I Know What You Did Last Summer: New Persistent Tracking Mechanisms in the Wild

STEFANO BELLORO[1] AND ALEXIOS MYLONAS[2], (Member, IEEE)
[1]British Broadcasting Corporation, London W1A 1AA, U.K.
[2]Department of Computing and Informatics, Bournemouth University, Poole BH12 5BB, U.K.

Corresponding author: Alexios Mylonas (amylonas@bournemouth.ac.uk)

ABSTRACT As the usage of the Web increases, so do the threats an everyday user faces. One of the most pervasive threats a Web user faces is tracking, which enables an entity to gain unauthorized access to the user's personal data. Through the years, many client storage technologies, such as cookies, have been used for this purpose and have been extensively studied in the literature. The focus of this paper is on three newer client storage mechanisms, namely, Web Storage, Web SQL Database, and Indexed Database API. Initially, a large-scale analysis of their usage on the Web is conducted to appraise their usage in the wild. Then, this paper examines the extent that they are used for tracking purposes. The results suggest that Web Storage is the most used among the three technologies. More importantly, to the best of our knowledge, this paper is the first to suggest Web tracking as the main use case of these technologies. Motivated by these results, this paper examines whether popular desktop and mobile browsers protect their users from tracking mechanisms that use Web Storage, Web SQL Database, and Indexed Database. Our results uncover many cases where the relevant security controls are ineffective, thus making it virtually impossible for certain users to avoid tracking.

INDEX TERMS Web tracking, web security, privacy, indexed database, indexedDB, web storage, Web SQL database.

I. INTRODUCTION

As of April 2018, the digital population has reached 4087 million users [1]. Most users access the web on a daily basis for the most diverse array of tasks, from sending emails and reading the news to browsing social media and accessing any kind of content. The usage of the Internet has improved the quality of our lives and provided us with opportunities and information, which were previously accessible only to a small percentage of people.

Nonetheless, such advantages do not come without a price. While users navigate the web, they expose themselves and share, willingly or not, personal information. Indeed, users are exposed to different threats, such as tracking and behavioral profiling, which directly violate their privacy. Many websites deploy a variety of technologies to track the users or profile them. These practices are used for a number of reasons [2]. For instance, identifying the user and knowing their characteristics enables a website to provide a more personalized user experience. While this may sound innocent and even desirable, the same techniques can be used to profile a possible target of a social engineering attack, gather personal information to either sell it, use it for advertising or for any other kind of surveillance [3]. Many client storage technologies have been used for tracking purposes over the years; the most famous of all is HTTP cookies.

Almost a decade ago, the web community was galvanized by the advent of HTML5 and the myriad of new primitive APIs associated to it. Among them, client-side storage APIs, such as Web Storage, Web SQL Database and Indexed Database API, were bound to revolutionize the web and eventually narrow the differences between web applications and native apps. Since then, the web has certainty evolved, but web applications are far from replacing native mobile apps. Moreover, in some instances, trackers have adopted client-side storage techniques as a way to enhance the capabilities of HTTP cookies, as shown by [35], but until now their use has been considered very limited.

In this context, this work focuses on Web Storage, Web SQL Database and Indexed Database API and investigates the usage of these client-side storage APIs as a tracking vector. Contrary to previous results in the literature, our results suggest that tracking is a major use case for these APIs. Moreover, we investigate the user control over the data that the aforementioned client-side technologies store on the user's device. Our results uncover multiple cases where the users are exposed to privacy violations, as: a) they are unable to delete data created by the API of Web Storage, Web SQL Database or Indexed Database API even though they are attempting to clear locally stored data of their browsing, and b) they unknowingly store potentially tracking data created by these APIs while browsing the web in a private session. These findings have serious privacy implications, as they highlight that it is virtually impossible for certain users to avoid web tracking.

Our contributions include:

- We perform a large-scale analysis of the usage of Web Storage, Web SQL Database or Indexed Database APIs on the web. We quantify their pervasiveness in the context of tracking code and find that these technologies are mostly used by trackers. To the best of our knowledge, we are the first to uncover that the main use case of these technologies is web tracking.
- We investigate the capability of modern, popular browsers for desktops and mobile devices to delete data that can be stored locally via these APIs. Moreover, we examine if data from these APIs remain after a private browsing session. In both cases, we find instances where the users would be exposed to privacy violations if a tracker uses Web Storage, Web SQL Database or Indexed Database APIs as the tracking vector, as we identified many cases that the relevant security control has questionable effectiveness.

The rest of the paper is organized as follows. Section II briefly provides the required background in client storage technologies. Section III investigates how frequently and for which purpose these APIs are used in the wild. Section IV reviews the controls offered to the users over these APIs. Finally, Section V presents the related work and Section VI concludes the paper and discusses future work.

II. BACKGROUND

Since the early days of the Web, HTTP cookies have been used as a client-side storage mechanism. As the web evolved, a desire for different and more capacious ways to store structured data on the web client started to emerge. Over the years, several client-based storage technologies appeared. Most of them, such as Local Shared object of Adobe Flash [10], Oracle Java [11], Microsoft Silverlight [12] and Google Gears (Google Code, 2008), were made available through third-party plug-ins. However, with the advent of HTML5, browsers started to support native functionalities that could replace these third-party plug-ins. Client-side persistent data storage technologies were introduced, such as Web Storage [13], Web SQL Database [15] and Indexed Database API [20]. This section briefly introduces the aforementioned three technologies, as well as cookies.

A. COOKIES

An HTTP cookie is a short piece of data (typically with size 4K) that a website sends to a client, either via HTTP response headers or by using client-side scripting. The client is expected to save this data and send it back to the server in subsequent HTTP requests. Each cookie is associated to an origin, *i.e.,* a combination of the hostname, the port number and the protocol used by the web application [5]. This is based on a concept known as '*same-origin policy*', which has been the cornerstone of browser security since the early days of the web [6].

For performance reasons, web browsers limit not only the length of HTTP cookies, but also apply constraints to their quantity, allowing only a few dozens per origin. Several online studies provide an overall view of the limits that different web browser vendors set to HTTP cookies [8], [9].

Since a webpage can contain resources from multiple origins, HTTP cookies are often used to identify and track users, not only across different browsing sessions, but also across different websites. Over the years, both Internet users and legislators have become more aware of the privacy implications of third-party tracking [7].

B. WEB STORAGE

Web Storage [13] is a specification that allows web applications to create a persistent key-value store in the browser, the content of which is maintained either until the end of a session (*i.e.,* sessionStorage), or beyond (*i.e.*, localStorage). This technology enables web applications to store a much greater amount of data compared to HTTP cookies. Specifically, the storage capacity provided by web storage varies from 5MB to 25MB, depending on the browser. An innovative feature of Web Storage is that a web application can use a client-side JavaScript API to retrieve locally stored data, even when the browser is offline. Web Storage is in fact completely based on client-side scripting and, unlike HTTP cookies, data cannot be sent via HTTP headers.

Similarly to HTTP cookies, the security model of Web Storage is based the same-origin policy. This means that each origin has a unique storage object assigned to it. For this reason, the specification does not recommended using this technology on websites that use a shared host name or do not use HTTPS. Otherwise, information leakage or spoofing may happen, as for example in the case of DNS spoofing attacks. Moreover, the specification recommends treating persistently stored data as potentially sensitive, as they could contain email addresses or calendar appointments, etc.

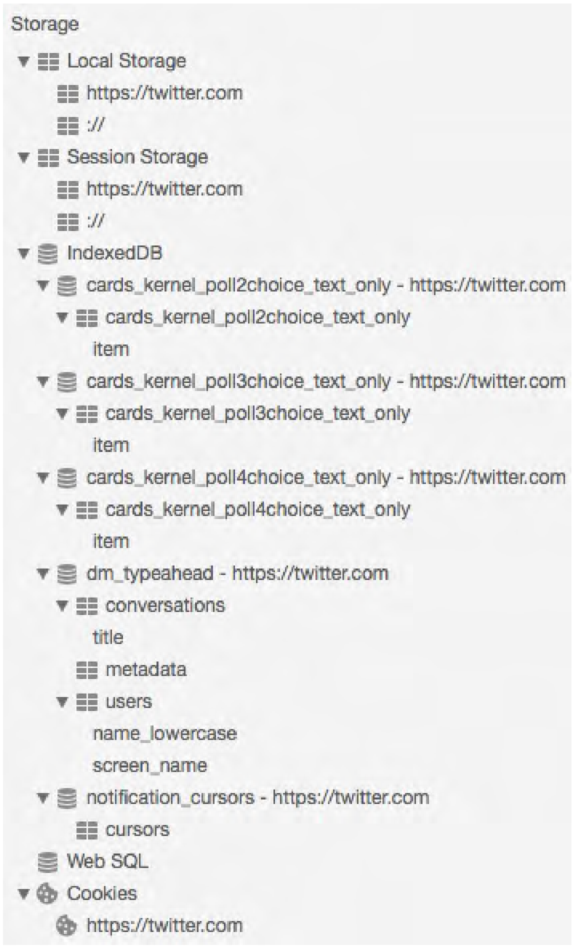
As with HTTP cookies, a third-party tracking agent could use Web Storage to profile users across multiple sessions [13]. The specification recommends browser vendors to treat web storage content in the same manner as they treat HTTP cookies. In particular, vendors are encouraged to organize the user interfaces for clearing data in a way that allows users to clear all different types of persistent data simultaneously. It is also important to point out that, while Web Storage is a much lesser known technology than HTTP cookies, its usage is not exempt from regulations around personal user data [14].

C. WEB SQL DATABASE

Web SQL Database [15] is a deprecated specification, which allows web applications to store large amounts of data in the browser, using client-side transactional databases that can be queried using SQL. The specification is based on SQLite, an embedded relational database management system developed by Owens [17]. Since the beginning of 2010, a few browser vendors started implementing experimental versions of the Web SQL database API [18]. This was not a complete novelty for some of them; Web SQL Database stores data in a very similar way to Google Gears and both technologies are based on SQLite. Other browser vendors like Mozilla [19], instead, decided to avoid Web SQL database completely. In November 2010, the W3C announced the decision to abandon the Web SQL Database draft lamenting the lack of multiple independent implementations. Web SQL Database was deprecated in favor of Indexed Database API. Despite the deprecation by the W3C, three major browser vendors (Chrome, Safari and Opera) have continued supporting Web SQL Database and have not yet announced any plan of discontinuing it.

D. INDEXED DATABASE API (INDEXED DB)

The first draft of this specification was initially published as WebSimpleDB API and it was renamed to Indexed Database API the following year [16]. It defines a JavaScript-based interface for an embedded transactional database system. Similarly to Web Storage and Web SQL Database, IndexedDB allows storing structured data in the browser and the API provided is the only interface a web application needs to access and manipulate them. The main difference with Web Storage is in the scale and structure of the data that can be stored. In fact, Web Storage provides a basic key-value store that can be useful when dealing with simple datasets. On the other hand, Indexed Database API enables the storage of larger amounts of structured data and provides advanced features, such as in-order key retrieval and storage of duplicate values for a key. Fig. 1 includes a snapshot from the console of Chrome that shows the client-side storage mechanisms, namely Web storage, IndexedDB, Web SQL and cookies, which are used by a Twitter Web application. It can be noted that IndexedDB can store data in a much more structured way compared to cookies and Web Storage, having several databases associated to the same origin. Each database has one or more object stores and their content can be sorted through one or multiple keys. Unlike Web SQL Database, IndexedDB is an object-oriented database. The interface for adding and retrieving data does not use SQL queries, but keys and indexes instead. The security recommendations for the usage of Indexed Database API are not different to those for Web Storage. The security model of IndexedDB still gravitates around the principles of the same-origin policy. A web application is allowed to access locally stored data as long as the request's origin matches the local database's origin. Unlike HTTP cookies, a maximum storage duration does not have to be specified.



```
Storage
▼ Local Storage
    https://twitter.com
    ://
▼ Session Storage
    https://twitter.com
    ://
▼ IndexedDB
  ▼ cards_kernel_poll2choice_text_only - https://twitter.com
    ▼ cards_kernel_poll2choice_text_only
        item
  ▼ cards_kernel_poll3choice_text_only - https://twitter.com
    ▼ cards_kernel_poll3choice_text_only
        item
  ▼ cards_kernel_poll4choice_text_only - https://twitter.com
    ▼ cards_kernel_poll4choice_text_only
        item
  ▼ dm_typeahead - https://twitter.com
    ▼ conversations
        title
      metadata
    ▼ users
        name_lowercase
        screen_name
  ▼ notification_cursors - https://twitter.com
      cursors
  Web SQL
▼ Cookies
    https://twitter.com
```

FIGURE 1. Representation of client-side stored data provided by the console of Chrome.

III. EXPLORING THE USAGE OF CLIENT-SIDE STORAGE IN THE WILD

This section discusses the methodology for investigating the usage of Web Storage, Indexed Database API and Web SQL Database as a tracking mechanism in the wild. In doing so, we first investigate the frequency of the usage of these technologies on a large-scale sample of the World Wide Web. Then, we quantify their pervasiveness in the context of third-party tracking code.

A. METHODOLOGY

In this subsection, we perform an analysis of a large-scale dataset, which contains snapshots of client-side scripts used by websites. The aim of our analysis is to demystify the pervasiveness of Web Storage, Indexed Database API and Web SQL Database in the web and study their use as a tracking vector. To this end, we perform static analysis on the dataset to identify instances of client-side scripts that make use of any of the three APIs by searching for code constructs that read and write data in the client. We then identify which of the above-mentioned scripts belong to well-known tracking domains. Fig. 2 shows a high-level diagram of our test environment.

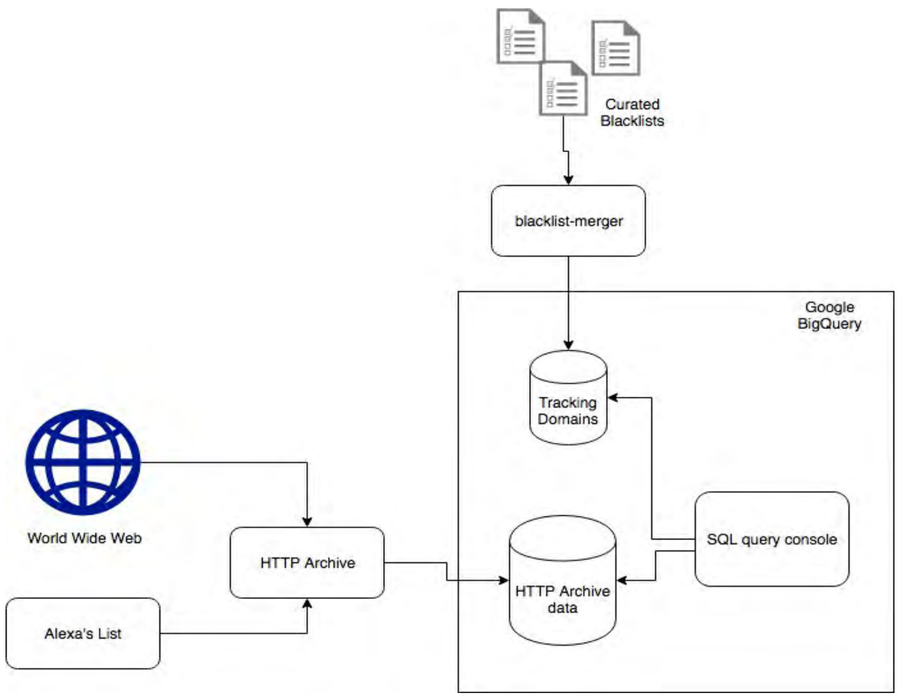


FIGURE 2. Architecture of the test environment.

The dataset in use comes from the HTTP Archive project created by [21]. Every fortnight, it crawls a list of webpages, which is loosely based on the Alexa Top Sites [22]. HTTP Archive collects data, such as the payload content and logs the interaction between the browser and the crawler. It also captures the body of the responses for each subresource (*i.e.* any file that is fetched by an HTML page such as scripts, stylesheets) used by the website. Since the size of the dataset generated by HTTP Archive can be up to several hundreds of gigabytes, Google BigQuery [23] was used for its processing.

For each of the three client-storage APIs one matching rule was used to create a series of SQL queries, which run against the HTTP Archive dataset using Google BigQuery. These rules, which are summarized in Table 1, were defined by using constructs required to perform basic operations, such as creating a data store, reading and writing data. Appendix A lays out the constructs that have been identified in this work in our matching rules.

TABLE 1. Matching rules used for each of api analyzed.

Primitive	Matching rule
Web Storage	"localStorage" AND ("setItem" OR "getItem")
IndexedDB	"indexedDB" AND "transaction" AND "objectStore"
Web SQL	"openDatabase" AND "transaction" AND "executeSql"

In order to identify whether a subresource belongs to a tracker, we created a database of tracking domains by aggregating three well-known tracking blacklists, namely: Disconnect (2017), No Track [26] and Easy List (2017). To this aim, we have developed scripts that combine the domains that are listed in the aforementioned blacklists after their files have been properly parsed and sanitized.

We run our experiments against: a) the whole dataset provided by HTTP Archive on the 15th of May 2018 and b) the Alexa top 10,000 sites. Table 2 summarizes the number of websites, subresources and truncated or empty subresources in our experiments. We highlight the low percentage of truncated or blank subresources, since on those the matching rules are not applicable.

TABLE 2. Data used from HTTP archive.

	Whole Dataset (May 2018)	Data matching Alexa's 10K sites
Number of websites in the dataset	460099	9020
Total number of subresources in the dataset	18860393	505745
Truncated or empty subresources (%)	3.15	5.26

B. EXPERIMENTAL RESULTS

Table 3 shows the usage of the primitives considered, on the whole dataset provided by HTTP Archive for the 15th of May 2018. An interesting result is that more than two thirds of the websites analyzed contain Web Storage related constructs. Another result worth noticing is that the constructs analysed are very often found on third party subresources. Similarly, Table 4, shows the results for the Alexa's top 10,000 sites. It is interesting to notice that in this case, the values for the usage of the Indexed Database API are

TABLE 3. Results for the whole dataset.

Client-side storage API	Websites with construct in subresource (%)	Websites with construct in 3rd party subresource (%)
Web Storage	71.66	65.39
IndexedDB	5.56	5.15
Web SQL DB	1.34	1.18

TABLE 4. Results for the alexa top 10K.

Client-side storage API	Websites with construct in subresource (%)	Websites with construct in 3rd party subresource (%)
Web Storage	83.09	77.08
IndexedDB	11.39	9.89
Web SQL DB	2.12	1.61

almost double compared to the whole dataset. The use of Web SQL remains low in our experiments, which is expected as this API is deprecated.

Table 5 summarizes the number of domains that include at least one tracking subresource, which is using one of the three client-side storage APIs. As it can be seen, there is a high percentage of websites containing at least one tracking subresource where constructs that belong to Web Storage (localStorage) can be found. The figures are much smaller for Indexed Database API and considerably smaller for Web SQL Database.

TABLE 5. Websites and tracking subresources.

API / Websites with at least one tracking subresource using API (%)	Whole Dataset (May 2018)	Data matching Alexa's 10K sites
Web Storage	57.72	67.21
IndexedDB	1.68	3.99
Web SQL DB	0.76	0.88

Finally, Table 6 highlights the usage of the client-side storage techniques in the context of tracking from a different angle. It shows amongst all the subresources that have been analysed, the percentage of them containing the constructs for the API considered that are used by a tracking domain. In other words, this table answers the question: "how frequently are those storage techniques used as tracking vectors?". In all cases, the frequencies are surprisingly high, starting from around 30% for Indexed Database API to more than 70% for Web Storage (localStorage). This significant finding suggests that currently user tracking is a major use case for the APIs that have been examined. Surprisingly, this is also the case for a deprecated standard, *i.e.,* Web SQL DB.

TABLE 6. Tracking subresources and primitives.

API/ Subresources using the API that are flagged as 'tracker' (%)	Whole Dataset (May 2018)	Data matching Alexa's 10K sites
Web Storage	71.18	63.88
IndexedDB	31.87	36.14
Web SQL DB	53.59	39.90

C. DISCUSSION

This section has shown that a significant number of the websites analysed contains at least one tracking subresource having code constructs that belong to at least one of the three APIs considered. More importantly, it has shown that tracking scripts seem to currently be the major use case of the three storage APIs considered. Indeed, in many cases, subresources that contain the analysed APIs are often identified as trackers. As our experiments used a dataset that represents a significant portion of the World Wide Web, we consider that our results shed some light on the usage of Web Storage, IndexedDB and Web SQL in user tracking.

However, the usage of HTTP Archive as the dataset for our experiments introduces a number of limitations to our work. HTTP Archive can only provide snapshots of front pages of openly available websites. The scanning engine does not perform operations such as user log in or following links on a menu. Considering that primitives such as the Indexed Database API are designed to support advanced web applications, it is reasonable to assume that there are cases of websites in which those storage techniques are used only once the user is logged in. However, this is an accepted limitation, especially considering that in order to quantify the usage of client-side storage techniques in the context of user tracking, it is far more important to focus on the large-scale adoption of the technologies in question rather than on specific use cases.

Another limitation of our work stems from the scanning engine of HTTP Archive, as it truncates payloads that are greater than 2 MBs. This means that if the constructs defined in the matching rules happen to be in the part of the payload that HTTP Archive could not capture, they will not be found by our queries. However, as shown in Table 2 truncation and empty subresources seldom appear in our dataset. Moreover, their absence does not invalidate our findings. On the contrary, their successful capture from HTTP Archive might provide additional subresources that match our rules, thus reinforcing our results.

In addition, HTTP Archive does not contain snapshots from each one of the Alexa Top one million sites. The set of websites scanned is loosely based on the Alexa list, but any private individual could send a request to HTTP Archive to add or remove sites to the dataset. The actual number of websites included in each scan is specified in the results section.

Finally, this work suffers from a limitation that is common in any static analysis approach. Our work verifies the presence of certain constructs in client-side scripts, but cannot verify the actual usage of the primitives unless the actual web application is executed in the browser, which falls outside the scope of our work. For example, a website could include a

JavaScript library that relies on Web Storage, but never execute its code in the browser. Moreover, some websites include third-party libraries that perform a set of basic operations using a given primitive with the sole purpose of assessing browser capabilities. This practice is known as 'feature detection' and one of the most well-known libraries used for this purpose is Modernizr [27].

IV. USER CONTROL OVER LOCALLY STORED DATA

The previous section uncovers that currently Web Storage, Indexed Database API and Web SQL Database are frequently used as a tracking vector. In this context, this section examines: i) whether popular desktop and smartphone browsers support the three aforementioned APIs, ii) the effectiveness of the deletion of the data stored by them as part of the mechanism that clears browsing data, and iii) if data remain when they are created in private browsing mode.

A. METHODOLOGY

As mentioned previously in section II.B, the specifications recommend browser vendors to treat the data removal of various client-side persistent data features in the same way as HTTP cookies. This means that browsers are expected to make it easy for users, or at least possible, to remove all locally stored user data. In addition, nowadays all browsers offer to their users the functionality to browse the web through a private session (often referred to as private or incognito mode). The primary aim of the private session is to allow users to browse the web without the browser saving data regarding the 'private' browsing history.

We built a simple web application, called Storage Watcher,[1] in order to verify the: a) level of API support in a given browser, and b) effectiveness of data deletion.

The tests were performed in June 2018, on a broad selection of desktop (Windows, Mac OS) and smartphone (Android, iOS, Windows Phone) browsers. These include the most popular browsers in these platforms, such as Firefox, Chrome, Safari, Opera, and Edge/Internet Explorer. Tables 11 and 12 in Appendix B include the details of the browsers that were analysed and the results of the abovementioned experiments.

B. EXPERIMENTAL RESULTS

Our results uncover inconsistencies with regards to the support of the client-side storage APIs by the different browsers (see Tables 11 and 12 in Appendix B). For example, amongst the desktop browsers, Firefox and Edge, disable the IndexedDB API when used in private browsing mode. In both cases, the other two storage APIs remain available. In contrast, certain versions of iOS WebKit-based browsers (Safari, Chrome and Firefox for iOS) and Firefox for Android, seem to do the exact opposite, as they disable the Web Storage and Web SQL Database APIs when in private mode, but not the IndexedDB API. It is, however, worth mentioning that more recent versions of iOS-WebKit-based browsers have introduced a more consistent approach on which all the three APIs are disabled on private browsing mode.

Our results also uncover multiple cases in which current popular browsers cannot protect the privacy of their users, as they fail to delete or isolate data stored via the API of Web Storage, Web SQL DB or IndexedDB. As summarized in Table 7 our results suggest that: a) the process of removing private data from a browser does not always delete data stored in all of the three client-side storage APIs or requires an extra step in the browser's user interface and b) some browsers do not fully isolate client-side stored data when used in private mode.

TABLE 7. Results for user control over local stored data.

Issue	OS	Browser	APIs
Data persists after clearing local data	iOS 10.2.1	Safari, Chrome 62.0	IndexedDB
	Android 6	Firefox 57, Firefox 60	IndexedDB
		MiuiBrowser 9.1.3	LocalStorage, IndexedDB
	Android 7	Firefox 54, Firefox 57	IndexedDB
	Android 8	Firefox 60	IndexedDB
Data deletion requires extra step in the UI	Windows Phone 8.10 by HTC	Internet Explorer	IndexedDB
	Mac OS 10.12.5	Firefox 57.0 (quantum), Firefox 56.0	IndexedDB
	Windows 10	Firefox 56	IndexedDB
	Windows XP	Firefox 47	LocalStorage, IndexedDB
		Firefox 56, 57	IndexedDB
Data persists after closing private session	iOS 11.1.2	Opera 16	LocalStorage
	Android 6	Opera 43.0	IndexedDB, Web SQL
		MiuiBrowser 9.1.3	LocalStorage, IndexedDB
	Android 7	Opera 42.7, Opera 43.0	IndexedDB, Web SQL
	Android 8	Opera 46.3	IndexedDB, Web SQL
Values from non-private session are leaked	Android 6	MiuiBrowser 9.1.3	IndexedDB
Data stored in guest mode is deleted only after quitting the browser	Mac OS 10.10.5, Windows 10	Chrome 62	localStorage, IndexedDB, Web SQL

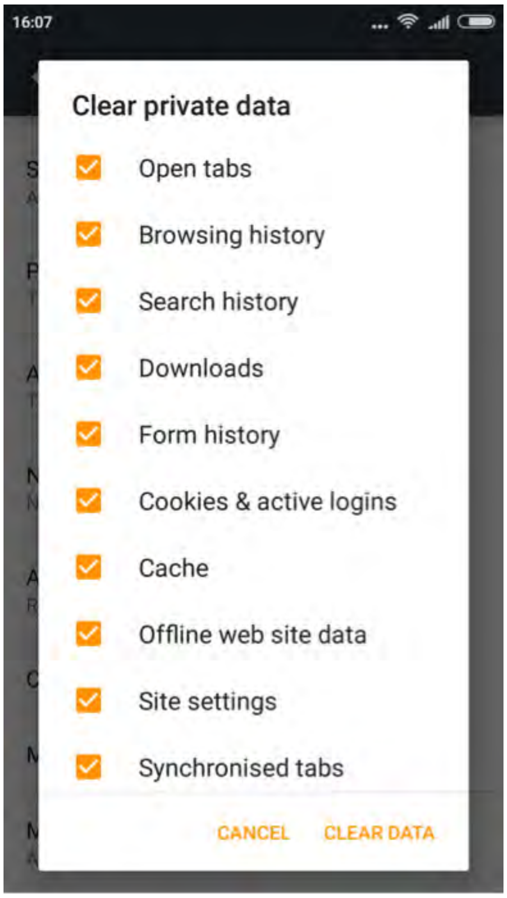
Specifically, certain versions of iOS-WebKit-based browsers (Safari[2] and Chrome for iOS[3]) and some Android

[1]Available at: https://github.com/stefano-belloro/storage-watcher

[2]Reported: https://bugs.webkit.org/show_bug.cgi?id=188164

[3]Reported https://bugs.chromium.org/p/chromium/issues/detail?id=868857

browsers (Firefox for Android[4] and MiuiBrowser) retain IndexedDB API content even after a user requests data deletion. In all the cases considered, the user interface not only does not make clear that IndexedDB API content will persist, but also gives the impression that all 'offline web site data' will be deleted (Fig. 3). Furthermore, in MiuiBrowser v.9.1.3, Web Storage (localStorage) content is also maintained, after a user requests the deletion of private data. Fortunately, in the case of iOS browsers, this issue seems to be resolved in the latest version of the software considered in this work. However, this behavior can still be seen on other recent browsers (i.e., Firefox 60 on Android 8).


16:07

Clear private data

- ✓ Open tabs
- ✓ Browsing history
- ✓ Search history
- ✓ Downloads
- ✓ Form history
- ✓ Cookies & active logins
- ✓ Cache
- ✓ Offline web site data
- ✓ Site settings
- ✓ Synchronised tabs

FIGURE 3. Firefox 57 on Android 6.0. The user interface suggest that offline data will be removed.

It is also worth pointing out that some browsers require the user to perform an extra action in order to include IndexedDB API content to the process of clearing private data. As a matter of fact, on all the desktop versions of Firefox[5] in scope of this work, whilst the user interface allows deleting data stored via IndexedDB API using the same panel used to remove HTTP cookies, this option is disabled by default. This means that users would have to expand the 'details' dropdown menu and manually add 'offline website data' if they wish to remove IndexedDB API content. On an earlier version of Firefox analysed (Firefox 47 on Windows XP), this was also the case for Web Storage (localStorage). This default setting could be misleading for an inexperienced user and give a sense of anonymity that cannot be guaranteed, especially considering that the IndexedDB API could be used as a backdoor to reinstate content of HTTP cookies [35].

Similarly, Internet Explorer for Windows Phone 8.10 by HTC requires a separate action to remove IndexedDB API content. In this case, the user needs to navigate to a different menu item called "advanced settings" and choose the option "manage storage".

Furthermore, Opera 43 on Android allows the persistence of data stored using IndexedDB API and Web SQL Database across different private browsing sessions.[6] Similarly, Opera for iOS exhibits the same behavior for Web Storage (localStorage) and MiuiBrowser 9.1.3 for both Web Storage (localStorage) and IndexedDB API.

Moreover, in Google Chrome's guest mode, content stored in each of the three APIs persists across different windows opened in guest mode.[7] This means that a user would need to quit Chrome completely in order to discard locally stored data accumulated in a guest browsing session. This behavior might be misleading for certain users who might assume that simply closing the browsing window but not the application might be enough to remove locally-stored private data.

Lastly, when running the experiment on MiuiBrowser 9.1.3, it was noticed that the browser carries over the values of IndexedDB API content created while using the application on normal browsing mode. As a result, if a private browsing session is preceded by a regular usage of the browser in its normal mode, MiuiBrowser allows a third party tracker to resume and recreate tracking values set while the user was browsing on previous non-private sessions and identify them even if they are browsing in private mode.

C. DISCUSSION

Our findings suggest that in many cases web users are exposed to privacy violations if the website they visit or any of its 3rd party subresources use Web storage, IndexedDB and Web SQL DB as a tracking vector. This holds true as our experiments uncovered instances in which: a) data persists after clearing local data or after closing a private session, b) data persists unless the user configures the browser appropriately, c) persistent data from a non-private session are leaked to the private session, and d) data stored in guest mode is deleted only after quitting Chrome. It is worth stressing, that non security and technically savvy users are more likely to use the default settings of the data clearing, thus failing to delete data that potentially violate their privacy in the cases that are describe in Table 7.

[4]Reported: https://bugzilla.mozilla.org/show_bug.cgi?id=1479403
[5]Reported: https://bugzilla.mozilla.org/show_bug.cgi?id=1479414
[6]Reported: Bug reference: DNAWIZ-38391
[7]Reported: https://bugs.chromium.org/p/chromium/issues/detail?id=868870

Our work also uncovers inconsistencies with regards to disabling certain client-storage APIs in private mode. If the reasoning for disabling the APIs is to prevent user tracking, it should be noted that advanced tracking mechanisms employ multi-tier approaches based on a combination of various storage vectors [35]. Therefore, blocking certain APIs whilst allowing the usage of others might not produce the desired level of privacy. Another interesting aspect is the way that browsers have implemented the security controls that handle the data of the APIs, namely private browsing and data clearing, is inconsistent across different versions of the same browsers and across different platforms (c.f. Table 11 and submitted bugs).

Moreover, our experiments include a) the most popular browsers of the popular operating systems for desktops (*i.e.,* Windows, Mac OS) and b) the most popular mobile browsers, which can be found in different types of mobile devices, such as smartphone and tablets, for the most popular platforms (*i.e.,* Android, iOS, Windows Phone). As these browsers currently hold the majority of the user share, we consider our results representative. Furthermore, as summarized in Table 7, it is worth noting that the majority of our findings concern popular mobile browsers, such as Chrome, Firefox and Safari. Given the popularity of these browsers and the fact that mobile devices are nowadays the primary vector to access the web [28], this increases the impact of our findings.

V. RELATED WORK

A. CLIENT-SIDE STORAGE SYSTEMS AS TRACKING VECTORS

Krishnamurthy and Wills [29] studied the diffusion of private user information performed by third-party trackers that use a combination of HTTP cookies and other elements of the DOM. The authors analysed a selection of 1200 popular websites and collected statistical data over a period of four years. The results showed that the collection of user data increased over time, even in websites where the user is expected to provide confidential information such as medical or financial details. More specifically, during the latest period that was analysed, September 2008, the penetration was 70%. Furthermore, it was discovered that 52% of the websites considered, contained code from at least two third-party tracking entities.

Gonzalez *et al.* [30] performed a large-scale study on the usage, content and format of HTTP cookies in the wild. Their work analysed a large dataset of network data that comprised of 5.6 billion HTTP requests. The authors determined the reach of cookies by measuring the number of referrers that generate an HTTP request to the same cookie-setting endpoint. They found that, while the vast majority of cookies relate to a unique referrer domain, there is a long tail of cookies whose originating requests come from a significantly high number of different domains. Moreover, the authors analysed the names of the cookies and found instances of websites that use cookies whose names include a unique identifier of the user. Finally, they discovered instances of cookies values containing personal identifiable information such as users' IP and email address, which, represent a serious breach of privacy.

Soltani *et al.* [31] conducted a study on the usage of Flash Local Shared Object, often referred to as 'Flash cookies', as a tracking vector. They analysed the top 100 domains ranked by QuantCast. On 31 of them, they found at least a case of data overlap between HTTP cookies and Flash cookies, meaning that the same value appeared on the data stored in both technologies. Moreover, they found several occurrences of what they defined as ''cookie respawning'', in which the value of a deleted HTTP cookie is restored in the background, taken from a Flash cookie that keeps its back up. On a follow-up study, Ayenson *et al.* [32] observed the emerging usage of Web Storage (localStorage) as a tracking vector. While the authors did not find if this storage system was directly employed as part of respawning mechanisms, they noticed several cases of matching values among HTTP cookies and Web Storage data, which they named 'HTML5 cookies'.

Roesner *et al.* [33] presented an in-depth investigation of web tracking performed by third-party actors. The work analysed a corpus of around 1000 websites, spanning from very popular to lesser-used websites, and found the presence of over 500 unique trackers. The authors proposed a classification of trackers that goes beyond the usual notion of first-party and third-party trackers. Instead, they introduced a classification system based on the tracking behavior that is observable from the client. This system challenges the significance of classifying cookies as either third-party or first-party. In fact, all cookies could be classified as first-party in the context of their own origins and often users visit those origins as 'first-party clients', such as in the case of social networks. For this reason, the authors suggested the usage of terms like ''tracker-owned'' cookies and ''site-owned'' cookies. The work also documented the occurrence of ''cookie leaks'', in which the contents of a cookie associated to a given origin are passed as parameters in a request to another origin, with the purpose of circumventing the browser's same-origin policy. Furthermore, the authors attempted to quantify the usage of alternatives to HTTP cookies. The authors found ''remarkably little use'' of Web Storage (localStorage). In fact, out of the 524 trackers identified, this storage mechanism was used in only 8 cases. Moreover, only 5 of them were found to contain unique identifies. All of those 5 cases were instances of cookie respawning, meaning that the user identifiers were copies of the values found on HTTP cookies. Finally, Flash LSOs were used by 35 trackers, but only 9 of them were identified as instances of cookie respawning.

Acar *et al.* [34] performed a large-scale analysis of a selection of advanced persistent tracking mechanisms. They reported the usage of Indexed Database API as a storage mechanism of tracking data, albeit in a small number of cases (20 out of the 100 000 analysed - 0.02%). The authors claimed to be the first to document evidence of the usage of IndexedDB as an evercookie vector. ''Evercookie'' is a technique that significantly increases the resilience of tracking HTTP cookies [35]. The mechanism consists of a client-side API that replicates the HTTP cookie data across several types of client-side storage systems.

Derksen *et al.* [36] also discussed the usage of Web Storage (localStorage) and Indexed Database API for tracking. The authors analysed the behavior of twenty popular tracking services on a selection of about a thousand websites. They found that localStorage was used by 15% of the trackers analysed. Moreover, none of the websites analysed showed the usage of Indexed Database API as a tracking vector. The authors also studied the implementation of data deletion. They found that the browsers they analysed allowed the deletion of both Web Storage (localStorage) and IndexedDB data, via the same mechanism that removes cookies. Similarly, Bujlow *et al.* [37], seem to imply that the content of data stored using these techniques is automatically emptied when the cookies are cleared. However, as this work uncovers currently in some popular browsers, data deletion requires either an extra step by the user in order to include HTML5-related client-side storage techniques or does not happen at all.

Another known practice used by trackers is cookie matching (or cookie syncing). This technique is used in real-time advertising bidding, allowing trackers to associate different tracking profiles that relate to the same user. Olejnik *et al.* [38] quantified both the frequency and the breadth of data leakage related to cookie matching. They analysed a sample of 100 user profiles and found that 91 of them were subject to cookie matching, showing instances of trackers leaking 27% of a user's browsing history. Moreover, they showed that the market value of parts of a users' browsing history can be as low as a fraction of a US dollar cent.

Englehardt [39] also discussed cookie-syncing, warning that it can allow the sharing of personal data between different tracking servers, without the user's direct consent. Cookie syncing can also further enhance the impact of cookie respawning. In fact, while most major trackers do not use mechanisms such as the aforementioned evercookie, they might share user information with trackers that do use techniques of cookie resurrection.

B. PREVENTIVE MEASURES AGAINST USER TRACKING

The 'Do Not Track' header was proposed by [40] as a measure against undesired user tracking. Compliant tracking agents are expected to refrain from identifying users and perform their usual activities according to the preference expressed by the user through the header. This proposal was extremely impactful and most major browser implemented the Do Not Track (DNT) header by the following year. Moreover, in 2015, the W3C started the work of formalizing this feature into a web standard called Tracking Preference Expression (DNT) [41].

However, according to Roesner et al. in [33], the 'Do Not Track' header does not seem to have any visible effect in preventing tracking, as it is a policy that relies on the goodwill of the tracker. Moreover, it appears that many of the parties involved with user tracking argue that their behavior should not be considered tracking as it is defined by the DNT specification, and consequentially refuse to implement it.

Furthermore, the authors pointed out that neither blocking third-party cookies is an effective method as some browsers only block the writing operation of a cookie, but not the reading. Therefore, the tracker would still be able to read the value of a cookie that has been set on a previous visit to social media sites or by advertising popups. Finally, the authors mentioned that private browsing mode is not an effective anti-tracking method because it is primarily designed to protect users from attackers with physical access to the machine and not necessarily from remote user tracking. As a method of protecting users' privacy, the authors propose ShareMeNot, a browser extension that limits third-party tracking code that belongs to social media sites, while making sure that actual functionality visible to the user remains unaffected. In practice, the extension allows tracking requests to be sent only when the user clicks on an embedded social media button (such as Facebook's "Like"). The solution proposed by the authors has been subsequently incorporated into another privacy tool named "Privacy Badger", a browser extension that uses algorithmic methods to decide which resource is tracking the user and verifies whether scripts that belong to a given domain collect unique identifiers even after sending

TABLE 8. Constructs used by web storage (localstorage).

Web Storage constructs	Usage
localStorage	Property of the 'window' object that needs to be used to access the Storage assigned to each origin
setItem	Method that adds a new item to the storage magnetic induction
getItem	Method that retrieves item to the storage

TABLE 9. Constructs used by indexed database API.

IndexedDB API constructs	Usage
indexedDB	Attribute of the 'window' object that provides applications a mechanism for accessing IndexedDB (of type 'IDBFactory')
transaction	Method needed to access the object store
objectStore	Method that returns an object store in the scope of the transaction

TABLE 10. Constructs used by web SQL database.

Web SQL Database constructs	Usage
openDatabase	Method that opens a Web SQL database, or creates a new one if none is found
transaction	Method to access the database
executeSql	Method that defines the SQL command to perform in a given transaction

TABLE 11. API support and data deletion results in the examined mobile browsers.

OS	Browser	Mode	API support			Data deletion		
			localStorage	IndexedDB	Web SQL	localStorage	IndexedDB	Web SQL
iOS 10.2.1	Safari	default	supported	Supported	supported	data deleted	***data persists after clearing local data***	data deleted
		private	***disabled***	Supported	***disabled***	N/A	data deleted	N/A
	Chrome 62.0	default	supported	Supported	supported	data deleted	***data persists after clearing local data***	data deleted
		incognito	***disabled***	Supported	***disabled***	N/A	data deleted	N/A
	Firefox 10.2	default	supported	Supported	supported	data deleted	data deleted	data deleted
		private	***disabled***	Supported	***disabled***	N/A	data deleted	N/A
	Opera 16	default	supported	supported	supported	data deleted	data deleted	data deleted
		private	supported	supported	supported	***data persists after closing private session***	data deleted	data deleted
		Mini	not supported	not supported	not supported	N/A	N/A	N/A
iOS 11.1.2	Safari	default	supported	supported	supported	data deleted	data deleted	data deleted
		private	***disabled***	***disabled***	***disabled***	N/A	N/A	N/A
	Firefox 10.3	default	supported	supported	supported	data deleted	data deleted	data deleted
		private	***disabled***	***disabled***	***disabled***	N/A	N/A	N/A
	Chrome 62.0	default	supported	supported	supported	data deleted	data deleted	data deleted
		private	***disabled***	***disabled***	***disabled***	N/A	N/A	N/A
	Opera 16	default	supported	supported	supported	data deleted	data deleted	data deleted
		private	supported	supported	supported	***data persists after closing private session***	data deleted	data deleted
		Mini	not supported	not supported	not supported	N/A	N/A	N/A
Windows Phone 8.10 by HTC	Internet Explorer	default	supported	supported	not supported	data deleted	***needs extra step: "advanced settings" > "manage storage"***	N/A
Android 6.0	Firefox 60.0.1	default	supported	supported	not supported	data deleted	***data persists after clearing local data***	N/A
		private	supported	not supported	not supported	data deleted	N/A	N/A
	Firefox 57	default	supported	supported	not supported	data deleted	***data persists after clearing local data***	N/A
		private	supported	not supported	not supported	data deleted	N/A	N/A
	Firefox Focus 2.4	default	supported	supported	not supported	data deleted	data deleted	N/A
	Chrome 66.0	default	supported	supported	supported	data deleted	data deleted	data deleted
		incognito	supported	supported	supported	data deleted	data deleted	data deleted
	Chrome 62.0	default	supported	supported	supported	data deleted	data deleted	data deleted
		incognito	supported	supported	supported	data deleted	data deleted	data deleted
	Opera 46.0	default	supported	supported	supported	data deleted	data deleted	data deleted
		incognito	supported	supported	supported	data deleted	data deleted	data deleted

TABLE 11. API support and data deletion results in the examined mobile browsers.

	Opera 43.0	default	supported	supported	supported	data deleted	data deleted	data deleted
		private	supported	supported	supported	data deleted	***data persists after closing private session***	***data persists after closing private session***
	Opera Mini 31.0	default	not supported	not supported	not supported	N/A	N/A	N/A
	Microsoft Edge Preview 1.0.0	default	supported	supported	supported	data deleted	data deleted	data deleted
		inPrivate	supported	supported	supported	data deleted	data deleted	data deleted
	MiuiBrowser 9.1.3	default	supported	supported	not supported	***data persists after clearing local data***	***data persists after clearing local data***	N/A
		incognito	supported	***carries over values from non incognito version***	not supported	***data persists after closing private session***	***data persists after closing private session***	N/A
	Edge 1.0	default	supported	supported	supported	data deleted	data deleted	data deleted
		inPrivate	supported	supported	supported	data deleted	data deleted	data deleted
Android 7.0	Firefox 57	default	supported	supported	not supported	data deleted	***data persists after clearing local data***	N/A
	Opera 43.0	private	supported	supported	supported	data deleted	***data persists after closing private session***	***data persists after closing private session***
Android 7.1	Chrome 65.0	default	supported	supported	supported	data deleted	data deleted	data deleted
		incognito	supported	supported	supported	data deleted	data deleted	data deleted
	Firefox Focus 5	default	supported	supported	not supported	data deleted	data deleted	N/A
	Opera 42.7	default	supported	supported	supported	data deleted	data deleted	data deleted
		private	supported	supported	supported	data deleted	***data persists after closing private session***	***data persists after closing private session***
	Firefox 54.0	default	supported	supported	not supported	data deleted	***data persists after clearing local data***	N/A
		private	supported	not supported	not supported	data deleted	N/A	N/A
Android 8.0	Chrome 66.0.3	default	supported	supported	supported	data deleted	data deleted	data deleted
		incognito	supported	supported	supported	data deleted	data deleted	data deleted
	Firefox Focus 5	default	supported	supported	not supported	data deleted	data deleted	N/A
	Opera 46.3	default	supported	supported	supported	data deleted	data deleted	data deleted
		private	supported	supported	supported	data deleted	***data persists after closing private session***	***data persists after closing private session***
	Firefox 60.0.1	default	supported	supported	not supported	data deleted	***data persists after clearing local data***	N/A
		private	supported	not supported	not supported	data deleted	N/A	N/A

a "Do Not Track" message. In this case, it automatically disallows content from that third-party tracker [42].

Mayer [43] studied a series of technologies developed to protect users from third-party trackers. The author found that community-maintained blacklists are the most effective way to prevent undesired user tracking. Those lists mainly consist of URLs or domains and are generally used in conjunction with browser extensions, such as AdBlock Plus [44]. The author also claimed that tracking is often inextricably tangled with third-party advertising, therefore often blocking trackers also entails blocking code that provides advertisements.

Mylonas *et al.* [45] analyzed the security controls of several mobile and desktop browsers. According to their results, desktop browsers generally provide better protection, as the

TABLE 12. API support and data deletion results in the examined desktop browsers.

OS	Browser	Mode	API support			Data deletion		
			localStorage	**IndexedDB**	**Web SQL**	**localStorage**	**IndexedDB**	**Web SQL**
Mac OS 10.12.5	Firefox 57.0 (quantum)	default	supported	supported	not supported	data deleted	***data deleted only if 'Offline website data' is explicitly selected by the user***	N/A
		private	supported	***disabled***	not supported	data deleted	N/A	N/A
	Firefox 56.0	default	supported	supported	not supported	data deleted	***data deleted only if 'Offline website data' is explicitly selected by the user***	N/A
		private	supported	***disabled***	not supported	data deleted	N/A	N/A
Mac OS 10.10.5	Chrome 62	guest	supported	supported	supported	***data deleted only after quitting chrome***	***data deleted only after quitting chrome***	***data deleted only after quitting chrome***
		default	supported	supported	supported	data deleted	data deleted	data deleted
		incognito	supported	supported	supported	data deleted	data deleted	data deleted
	Opera 49	default	supported	supported	supported	data deleted	data deleted	data deleted
		private	supported	supported	supported	data deleted	data deleted	data deleted
	Safari 10.1.1	default	supported	supported	supported	data deleted	data deleted	data deleted
		private	***disabled***	supported	***disabled***	N/A	data deleted	N/A
Windows 10	Edge 40	default	supported	supported	not supported	data deleted	data deleted	N/A
		inPrivate	supported	***disabled***	not supported	data deleted	N/A	N/A
	Chrome 62	default	supported	supported	supported	data deleted	data deleted	data deleted
		guest	supported	supported	supported	***data deleted only after quitting chrome***	***data deleted only after quitting chrome***	***data deleted only after quitting chrome***
		incognito	supported	supported	supported	data deleted	data deleted	data deleted
	Firefox 56	default	supported	supported	not supported	data deleted	***data deleted only if 'Offline website data' is explicitly selected by the user***	N/A
		private	supported	***disabled***	not supported	data deleted	N/A	N/A
	Opera 49	default	supported	supported	supported	data deleted	data deleted	data deleted
		private	supported	supported	supported	data deleted	data deleted	data deleted
Windows XP	Internet Explorer 11	default	supported	supported	not supported	data deleted	data deleted	N/A
	Chrome 62	default	supported	supported	supported	data deleted	data deleted	data deleted
		incognito	supported	supported	supported	data deleted	data deleted	data deleted
	Firefox 47	default	supported	supported	not supported	***data deleted only if 'Offline website data' is explicitly selected by the user***	***data deleted only if 'Offline website data' is explicitly selected by the user***	N/A

controls available on them perform better than those available on their mobile counterparts. For example, users of the mobile browsers do not have the option to opt-out of third-party cookies and in many cases the interface that allows the user to control security features can be confusing. Finally, the authors found a number of security issues on two major mobile browsers and also pointed out that in most of the mobile browsers the 'Do Not Track' header is unavailable.

Virvilis *et al.* [46] compared the different protection measures against rogue sites offered by desktop and mobile browsers. According to their results mobile browsers often offer a lower level of protection compared to their desktop-based counterparts and in some cases they offer no protection at all. Furthermore, the authors introduced Secure Proxy, a new browser-independent countermeasure that overcomes the technical limitations related to each specific browser without the need of browser extensions. Secure Proxy consists of a HTTP forward proxy that operates at network level to filter content before it reaches the user's device. The filtering mechanism is delegated to a third-party service that assesses the reliability of the content providers, based on the aggregation of multiple blacklists and Antivirus engines.

Building from the previous work, Nisioti *et al.* [47] revisit the anti-phishing mechanisms available for users of mobile browsers of three popular operating systems. The study revealed that the protection provided by pre-installed web browsers is still very poor and in most cases non-existent. The only browsers that offer an adequate level of protection are Firefox and Chrome on Android. Moreover, in iOS, neither the default browser nor any of the third-party browsers offer any protection against phishing attacks. In this context, the authors proposed TRAWL (TRAnsparent Web protection for alL), an extension of 'Secure Proxy'. Similarly to 'Secure Proxy', TRAWL is implemented outside the users' device in order to avoid resource consumption and to offer cross platform compatibility. The tool provides DNS and URL filtering based on a collection of curated blacklists, but instead of delegating the filtering to a third-party service it performs it locally. In this way, the user's privacy is preserved and any third party limitations are overcome.

Similarly, Kontaxis and Chew [48] present a new anti-tracking mechanism of Mozilla Firefox, called Tracking Protection. The mechanism is similar to ad-blocking browser extensions such as AdBlock Plus. It analyses all outgoing HTTP requests and matches them against a blacklist, which is based on a curated list of tracking origins. The authors evaluated their approach against 200 popular news sites and according to the results there was a 67.5% reduction in the number of HTTP cookies. Moreover, this approach resulted on a 44% median reduction in page load time and 39% reduction in data usage for the testes sites.

VI. CONCLUSION

Online tracking is an everyday practice and, when it is performed against the user's will it is a major privacy violation. While older client-side storage technologies such as cookies have been studied extensively as tracking vectors, newer technologies, *i.e.,* Web Storage, Indexed Database API and Web SQL Database, have not received the same level of attention. In this paper, we measure the frequency of use of these technologies on a HTTP Archive dataset, which constitutes a representative sample of the World Wide Web, and examine the extent to which they are used for tracking purposes. As shown by the results, currently there is a large fraction of websites that utilize the three primitives, with Web Storage being the most used. However, the most alarming result is the frequency in which these APIs seem to used by trackers, which for all three technologies seems to be higher than 30% and in particular almost 70% for Web Storage. Finally, we examined whether the current popular web browsers for desktops and mobile devices can protect their users from privacy violations that use the aforementioned three technologies as the tracking vector. Our results suggest that in many cases the relevant security controls (*i.e.,* data clearing and private mode) are ineffective in deleting the relevant data and ensuring isolation of the data when used in private sessions. The bugs that were identified in this work have been reported to the relevant browser vendors as indicated in section 4.B.

APPENDIX A
MATCHING RULES USED IN STATIC ANALYSIS

The Web Storage API provides two storage mechanisms, one for handling data within a current session (sessionStorage) and another one that lasts beyond the current session (localStorage). In this work, only the constructs used by localStorage were considered, as content stored using sessionStorage expires at the end of a browsing session. TABLE 8 shows the constructs needed in order to read or write data using localStorage.

The same process was followed for the Indexed Database API. The constructs mentioned in TABLE 9 are part of the steps necessary to create a local database containing an object store and to access the store to either read or write data.

Similarly, Table 10 shows the constructs necessary to read and write data using the now deprecated Web SQL Database API.

APPENDIX B
FULL RESULTS OF SECTION IV

Tables 11 and 12 provide all the results from the experiments that were described, summarized and discussed in Section IV.

REFERENCES

[1] *Types of Personal Information and Images Shared Digitally by Global Internet Users as of January 2017*. Accessed: Jan. 2018. [Online]. Available: https://www.statista.com/statistics/266835/sharing-content-among-us-internet-users/

[2] C. Castelluccia and A. Narayanan, "Privacy considerations of online behavioural tracking," Eur. Union Agency Netw. Inf. Secur., Heraklion, Greece, Tech. Rep. Deliverable–2012-10-19, 2012.

[3] S. Englehardt *et al.*, "Cookies that give you away: The surveillance implications of Web tracking," in *Proc. 24th Int. Conf. World Wide Web*, May 2015, pp. 289–299

[4] T. Bujlow, V. Carela-Español, J. Solé-Pareta, and P. Barlet-Ros. (2015). "Web tracking: Mechanisms, implications, and defenses." [Online]. Available: https://arxiv.org/pdf/1507.07872.pdf

[5] A. Barth. (2011). *The Web Origin Concept 2011 IETF RFC6454*. [Online]. Available: https://tools.ietf.org/html/rfc6454

[6] E. Shepherd. (2017). *Same-Origin Policy in MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

[7] D. M. Kristol, "HTTP cookies: Standards, privacy, and politics," *ACM Trans. Internet Technol.*, vol. 1, no. 2, pp. 151–198, 2001.

[8] J. Manico. (2009). *Real World Cookie Length Limits*. Manicode. [Online]. Available: http://manicode.blogspot.hk/2009/08/real-world-cookie-length-limits.html
[9] I. Roberts. (2013). *Browser Cookie Limits*. [Online]. Available: http://browsercookielimits.squawky.net/
[10] Adobe Systems. (2012). *What Are Local Shared Objects?* Security and Privacy. [Online]. Available: http://web.archive.org/web/20121230094342/http://www.adobe.com/security/flashplayer/articles/lso/
[11] Oracle. (2017). *Java Documentation*. [Online]. Available: http://docs.oracle.com/en/java
[12] Microsoft. (2017). *What is Silverlight?*. [Online]. Available: https://www.microsoft.com/silverlight/what-is-silverlight/default
[13] Web Hypertext Application Technology Working Group. (2017) *Web Storage in HTML Living Standard*. [Online]. Available: https://html.spec.whatwg.org/multipage/webstorage.html
[14] European Commission. *Cookies European Commission*. Accessed: Jan. 2018. [Online]. Available: http://ec.europa.eu/ipg/basics/legal/cookies/index_en.htm
[15] I. Hickson. (2010). *Web SQL Database. W3C Working Group Note 18 November 2010*. Accessed: Jan. 2018. [Online]. Available: https://www.w3.org/TR/2010/NOTE-webdatabase-20101118
[16] N. R. Mehta. (2009). *WebSimpleDB, A.P.I., in W3C Working Draft*. [Online]. Available: https://www.w3.org/TR/2009/WD-WebSimpleDB-20090929
[17] M. Owens, *Introducing SQLite. The Definitive Guide to SQLite*. New York, NY, USA: Apress LP, 2006, pp. 1–16.
[18] Chromium Blog. (2010). *More Resources for Developers*. [Online]. Available: https://blog.chromium.org/2010/01/more-resources-for-developers.html
[19] A. Ranganathan. (2010). *Beyond HTML5: Database APIs and the Road to IndexedDB. Mozilla Hacks*. Accessed: Jan. 2018. [Online]. Available: https://hacks.mozilla.org/2010/06/beyond-html5-database-apis-and-the-road-to-indexeddb
[20] A. Alabbas and J. Bell. (2017). *Indexed Database API 2.0, W3C Proposed Recommendation*. Accessed: Nov. 16, 2017. [Online]. Available: https://www.w3.org/TR/IndexedDB-2
[21] S. Sounders. (2011), *Announcing the HTTP Archive, High Performance Web Sites Blog*, https://www.stevesouders.com/blog/2011/03/30/announcing-the-http-archive/
[22] Alexa Internet, Inc. (2017), *Alexa Top 1,000,000 Sites*. [Online]. Available: http://s3.amazonaws.com/alexa-static/top-1m.csv.zip
[23] I. Grigorik. (2013). *HTTP Archive + BigQuery = Web Performance Answers, in Author's Blog*. [Online]. Available: https://www.igvita.com/2013/06/20/http-archive-bigquery-web-performance-answers/
[24] Google Cloud Platform. (2017). *SQL Reference*. [Online]. Available: https://cloud.google.com/bigquery/docs/reference/standard-sql/
[25] *Information Technology—Database Languages—SQL—Part 11: Information and Definition Schemas, (SQL/Schemata)*, standard ISO/IEC 9075-11:201, International Organization for Standardization IEC JTC 1/SC 32, 2011. [Online]. Available: https://www.iso.org/standard/53685.html
[26] Quidsup. (2017). *NoTrack*. [Online]. Available: https://github.com/quidsup/notrack
[27] F. Ateş. (2017). *What is Modernizr?* [Online]. Available: https://modernizr.com/docs/#what-is-modernizr
[28] R. V. D. Meulen and C. Pettey. (2012). *Gartner Survey Highlights Top Five Daily Activities on Media Tablets*. [Online]. Available: https://www.gartner.com/newsroom/id/2070515
[29] B. Krishnamurthy and C. Wills, "Privacy diffusion on the Web: A longitudinal perspective," in *Proc. 18th Int. Conf. World wide web*, 2009, pp. 541–550.
[30] R. Gonzalez *et al.*, "The cookie recipe: Untangling the use of cookies in the wild," in *Proc. IEEE Netw. Traffic Meas. Anal. Conf. (TMA)*, Jun. 2017, pp. 1–9.
[31] A. Soltani, S. Canty, Q. Mayo, L. Thomas, and C. J. Hoofnagle, "Flash cookies and privacy," in *Proc. AAAI Spring Symp., Intell. Inf. Privacy Manage.*, Mar. 2010, pp. 158–163.
[32] M. D. Ayenson, D. J. Wambach, A. Soltani, N. Good, and C. J. Hoofnagle. (2011). *Flash Cookies and Privacy II: Now With HTML5 and ETag Respawning*. [Online]. Available: https://www.truststc.org/education/reu/11/Posters/AyensonMWambachDpaper.pdf
[33] F. Roesner, T. Kohno, and D. Wetherall, "Detecting and defending against third-party tracking on the Web," in *Proc. 9th USENIX Conf. Netw. Syst. Design Implement.*, Apr. 2012, pp. 1–12.
[34] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The Web never forgets: Persistent tracking mechanisms in the wild," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2014, pp. 674–689.
[35] S. Kamkar. (2010). *Evercookie*. [Online]. Available: http://samy.pl/evercookie
[36] I. Derksen, I. E. Poll, and F. van den Broek. (2016). *HTML5 Tracking Techniques in Practice*. [Online]. Available: http://www.cs.ru.nl/bachelorscripties/2016/Ivar_Derksen___4375408___HTML5_Tracking_Techniques_in_Practice.pdf
[37] T. Bujlow, V. Carela-Español, J. Solé-Pareta, and P. Barlet-Ros. (2015). *Web Tracking: Mechanisms, Implications, and Defences*. [Online]. Available: https://arxiv.org/abs/1507.07872
[38] L. Olejnik, T. Minh-Dung, and C. Castelluccia. (2013). *Selling off Privacy at Auction*. [Online]. Available: https://hal.inria.fr/hal-00915249
[39] S. Englehardt. (2014). The hidden perils of cookie syncing. Freedom to Tinker. [Online]. Available: https://freedom-to-tinker.com/2014/08/07/the-hidden-perils-of-cookie-syncing/
[40] C. Soghoian. (2011). *Slight Paranoia: The History of the do not Track Header*. Accessed: Jan. 2018. [Online]. Available: http://paranoia.dubfire.net/2011/01/history-of-donot-track-header.html
[41] R. Fielding and D. Singer. (2017). *Tracking Preference Expression (DNT)*. [Online]. Available: https://www.w3.org/TR/tracking-dnt/
[42] (2017). *Privacy Badger*. [Online]. Available: https://www.eff.org/privacybadger
[43] J. Mayer. (2011). *Tracking the Trackers: Self-Help Tools. The Center for Internet & Society*. [Online]. Available: http://cyberlaw.stanford.edu/blog/2011/09/tracking-trackers-self-help-tools
[44] Eyeo GmbH. (2017). *Getting Started with Adblock Plus*. [Online]. Available: https://adblockplus.org/getting_started#general
[45] A. Mylonas, N. Tsalis, and D. Gritzalis, "Evaluating the manageability of Web browsers controls," in *Proc. Int. Workshop Secur. Trust Manage.* Berlin, Germany: Springer, Sep. 2013, pp. 82–98.
[46] N. Virvilis *et al.*, "Security Busters: Web browser security vs. rogue sites," *Comput. Secur.*, vol. 52, pp. 90–105, 2015.
[47] A. Nisioti, M. Heydari, A. Mylonas, V. Katos, and V. H. F. Tafreshi, "TRAWL: Protection against rogue sites for the masses," in *Proc. 11th Int. Conf. Res. Challenges Inf. Sci. (RCIS)*, May 2017, pp. 120–127.
[48] G. Kontaxis and M. Chew. (2015). "Tracking protection in Firefox for privacy and performance." [Online]. Available: https://arxiv.org/abs/1506.04104

STEFANO BELLORO received the M.Sc. degree in software engineering and Internet architecture with a dissertation in cybersecurity. He was leading the Web teams, where he was responsible for the BBC World Service Web portfolio, providing news in more than 40 different languages. He is currently a Software Engineering Manager with BBC. He also looks after the teams that build and support software and tools for broadcasting.

ALEXIOS MYLONAS (M'09) received the B.Sc. degree (Hons.) in computer science from the Athens University of Economics and Business, the M.Sc. degree in information security from the Royal Holloway, University of London, and the Ph.D. degree in information and communication security from the Athens University of Economics and Business. He was a Security Consultant with VeriSign's PKI Trust Network. He was a Lecturer with Staffordshire University. He is currently a Lecturer with Bournemouth University. He is also an expert in cybersecurity. He has more than 20 publications that are well referenced and appear in esteemed conference and journal publications. His current research interests include cybersecurity, threat intelligence, and Web security. He is also a member of ACM. He has served as a technical committee member for conferences and journals.