

BACHELOR THESIS  
COMPUTER SCIENCE



RADBOUD UNIVERSITY

---

# HTML5 Tracking Techniques in Practice

---

*Author:*  
Ivar Derksen  
4375408

*First supervisor/assessor:*  
Dr. Ir. Erik Poll  
e.poll@cs.ru.nl

*Second assessor:*  
Fabian van den Broek, MSc  
f.vandenbroek@cs.ru.nl

July 7, 2016

## **Abstract**

In this research we look into tracking techniques available in the HTML5 standard and how these techniques are used in practice. In here we focus on the HTML5 storage mechanisms Local Storage and IndexedDB.

We found out that the privacy recommendations mentioned in the HTML5 specification do not prevent user tracking. The measures are similar to those of HTTP cookies. Browser manufacturers did also not implement sufficient privacy measures on their own initiative. The extra tracking risks for users do not increase much, because when HTML5 is used HTTP cookies are also available to use for tracking parties. It only creates more potential for real-time user tracking on web pages.

We also developed a browser extension for Google Chrome to be able to measure which tracking parties use the HTML5 storage techniques Local Storage and IndexedDB. The results of the data collection using this extension showed that a few tracking parties use Local Storage for tracking purposes. IndexedDB is hardly used anywhere for this purpose.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Background &amp; Related Work</b>	<b>5</b>
2.1	Storage Based Tracking Techniques . . . . .	7
2.1.1	HTTP Cookies . . . . .	7
2.1.2	Flash cookies . . . . .	7
2.1.3	Other . . . . .	7
2.2	Cache Based Tracking Techniques . . . . .	8
2.3	Fingerprint Based Tracking Techniques . . . . .	8
2.3.1	Canvas Fingerprinting . . . . .	8
2.4	Definitions . . . . .	9
2.4.1	Super Cookies . . . . .	9
2.4.2	Evercookies . . . . .	9
2.5	Related Work about Empirical Analysis . . . . .	10
<b>3</b>	<b>Tracking Techniques in HTML5</b>	<b>11</b>
3.1	Local Storage . . . . .	12
3.1.1	Functioning . . . . .	13
3.2	IndexedDB . . . . .	13
3.2.1	Functioning . . . . .	13
3.3	Tracking Method . . . . .	17
<b>4</b>	<b>Privacy Measures in HTML5</b>	<b>19</b>
4.1	Tracking Prevention Recommendations W3C . . . . .	19
4.2	Comparison: Tracking Prevention in Browsers . . . . .	20
4.2.1	Testing Set-up . . . . .	20
4.2.2	Tests . . . . .	21
4.3	Comparison: Protection Against Cross Frame Communica- tion in Browsers . . . . .	26
4.3.1	Parameters in Source URL . . . . .	26
4.3.2	Cross Frame DOM Access . . . . .	27
4.3.3	postMessage API . . . . .	27

<b>5</b>	<b>Empirical Analysis of HTML5 Tracking Techniques</b>	<b>28</b>
5.1	Orientation phase . . . . .	28
5.2	Method . . . . .	29
5.2.1	Chosen Tracking Parties . . . . .	30
5.2.2	Chosen Websites to Analyse . . . . .	31
5.2.3	Browser Extension for Collecting Results . . . . .	31
5.3	Results . . . . .	33
5.3.1	Local Storage Usage . . . . .	33
5.3.2	IndexedDB Usage . . . . .	34
<b>6</b>	<b>Future Work</b>	<b>35</b>
<b>7</b>	<b>Conclusions</b>	<b>37</b>
<b>A</b>	<b>Example: Data Storing with IndexedDB</b>	<b>43</b>
<b>B</b>	<b>Visited Websites for Empirical Analysis</b>	<b>45</b>
<b>C</b>	<b>Chrome Extension Source Code</b>	<b>46</b>
C.1	check_current_frame.js . . . . .	46
C.2	background.js . . . . .	47
C.3	popup.js . . . . .	49
<b>D</b>	<b>Testing Results of Empirical Analysis (Digital Version Only)</b>	<b>51</b>

# Chapter 1

## Introduction

In the beginning of the Internet era websites were static pages without any interaction. Since then it evolved in becoming more and more complex and dynamic. Therefore all kind of extensions and plugins were introduced to make this possible. These methods also took risks with them for users. It made it possible to collect information about the web behaviour of people. In most cases this was not the intention of the developers.

A lot of research has been done in discovering new techniques to track users. Tracking is not completely dependent of traditional HTTP cookies on its own. There are several storing techniques built-in in plugins like Adobe Flash, Microsoft Silverlight or HTML5. There are also other browser features that are not intentionally created for storage that can be misused. Here you can think of methods like canvas fingerprinting or misusing the image and file cache mechanisms. These tracking techniques are often called *super cookies*. Several of these techniques can be combined to create a tracking mechanism that is difficult to beat. Therefore this combination method is often referred to as *evercookies*. In Chapter 2 some well known techniques will be explained.

Thus, now we know many techniques are present, but we do not know for sure what is really used in practice nowadays. The advertising world moves quickly. Tracking techniques rise and fall, causing that research of only a few years old might be outdated already. Advertising networks are in a constant fight to collect as much data as possible. People more and more try to avoid these practices by for example using ad blockers or cookie blockers. For example, around 198 million people in the world use ad blockers [25]. When we zoom in to the Netherlands, we can see that around 2.2 million browsers that actively use an ad blocker [25]. These numbers are based on the number of downloads of the software and how often updates for block lists are downloaded. The trend that more people use ad and cookie blockers affects business models of advertising companies. Therefore it is likely that they try to find new techniques to keep one step ahead.

The field of web standards is recently extended with the introduction of HTML5 [13]. This new specification includes several techniques to make more complex web applications without the need of browser plugins. From a privacy perspective this development in itself is good, because it reduces the number of tools that may harm your privacy. HTML in general, so also HTML5, are usually used via a HTTP connection. This means that HTTP cookies are still available as an alternative tracking technique.

We would like to know: what is the current state of online tracking since the introduction of HTML5. To determine this we will look into how much these techniques are used in practice, what potentials it offers to track users and eventually how users can prevent themselves against them.

Because the HTML5 specification contains a large variety of features that may have potentials to be used as tracking technique, we decided to focus on the storage techniques included in the HTML5 specification: Local Storage [12] and IndexedDB [17].

We expect that storage techniques included in HTML5 do not prevent much better against user tracking than traditional HTTP cookies do. Without any protocol change HTTP tracking techniques will remain possible with HTML5, because HTML pages are usually accessed using HTTP. Therefore the developers of HTML5 techniques may not be motivated to improve privacy in a higher layer of abstraction, because advertising networks will then fall back on the techniques included in underlying protocols.

In Chapter 2 we give an overview of well known tracking techniques other than the storage techniques included in HTML5. We also discuss other related work to this research here. The particular storage techniques Local Storage and IndexedDB included in the HTML5 specifications are discussed in Chapter 3. We also explain here how tracking would look like using these techniques.

In Chapter 4 we discuss what privacy measures are taken to prevent user tracking using Local Storage and IndexedDB.

Chapter 5 is about the usage of Local Storage and IndexedDB for tracking in practice. In here an overview is given of several possible methods to do empirical analysis of tracking methods. Furthermore we discuss the browser extension “Tracking Trackers” we made with which we did the empirical analysis and we discuss the results we generated with it.

## Chapter 2

# Background & Related Work

Throughout the years a large collection of mechanisms was formed. These mechanisms have the potential to track the browsing behaviour of users. In 2015 Bujlow *et al.* [5] made an overview of the most well-known tracking techniques. To structure this overview Bujlow *et al.* [5] made a classification. With this we can better distinguish all kinds of methods. In this background section we will therefore use this classification. In Section 2.4 the most important other methods to structure known methods will be mentioned. Finally, we discuss other related work about empirical analysis of tracking usage in Section 2.5.

In general Bujlow *et al.* [5] distinguish three major types of tracking techniques:

- Storage-based tracking techniques (Section 2.1)  
Bujlow *et al.* [5] define storage-based mechanisms as mechanisms where some identifying string is saved.
- Cache-based tracking techniques (Section 2.2)  
These techniques work rather similar to storage based techniques, but here not the identifier itself is stored, but a user can be identified by the availability of cached files or information that it stored in metadata of files.
- Fingerprint-based tracking techniques (Section 2.3)  
The final category covers all techniques that tend to recognize a user on information he leaks. In other words: it is not based on identifying information that is delivered by the one that wants to track people and stored somewhere by the client. The information is generated or delivered by the client's device itself. Usually tracking cannot be done using only one piece of information, because that is not unique enough. Therefore multiple techniques can be combined to make sure there is sufficient entropy.

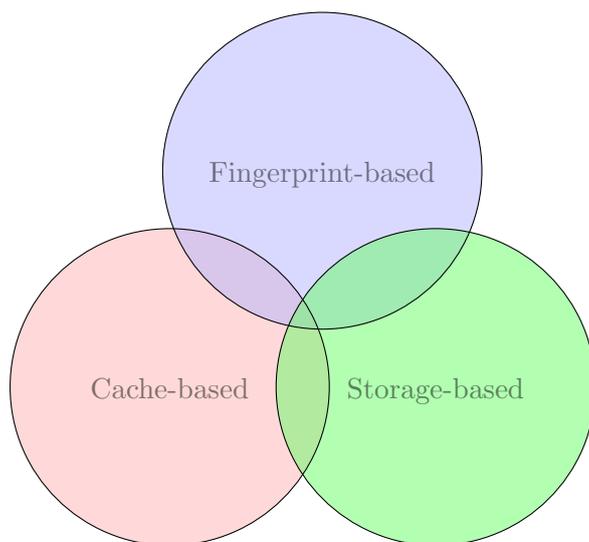


Figure 2.1: Outline tracking techniques categories

By this distinction must be noted that Bujlow *et al.* [5] also have a category with session-only techniques. These can be used to authenticate a user or to track user behaviour within one particular session of a site. Techniques that can be classified in this category are for example session-only HTTP cookies, values saved in JavaScript variables and HTML5's Session Storage. These techniques cannot be used by itself to track users across multiple visits, so especially not across multiple websites. Therefore session-only techniques are not that relevant in this case.

Not all tracking techniques can be easily categorized in one of the three types. Sometimes a technique has characteristics of multiple types. For example one could imagine a fingerprinting technique that only works if some files are cached. This means there is some overlap between the multiple types. A sketch of the tracking technique landscape would be like the diagram in Figure 2.1.

Furthermore multiple tracking techniques can be combined to make a better working entirety. With this flaws of certain techniques can be compensated with advantages of other techniques. This will make tracking difficult to circumvent for users.

In this chapter an overview is given of well-known tracking techniques that are not directly involved in this research. The specific HTML5 tracking methods that we will examine more deeply are discussed in Chapter 3.

## 2.1 Storage Based Tracking Techniques

### 2.1.1 HTTP Cookies

The traditional way to save information locally in a browser is using HTTP cookies [4]. Because HTTP itself is stateless this technique is necessary to make dynamic web pages. Due to the *Same Origin Policy* [28] cookies can only be accessed by all websites which URL ends in the same URL as the website that set it. However, this system does not prevent that parties track you across multiple websites, because if a websites refers to some content of third party's site, then that site is allowed to read its own cookies again.

There are different kind of cookies [4]. In the first place there are persistent and non-persistent HTTP cookies. Non-persistent cookies are only stored during one browser session. When the browser is closed the cookies should be deleted again. Persistent cookies are stored permanently. Cookies do have an expiry date to prevent that cookies can be set forever.

Both types of cookies can also have a HTTP-only and a secure flag set. HTTP-only cookies can only be accessed for the direct HTTP responses to the corresponding server. For example, they cannot be read by running a script in the browser. Secure cookies can only be send if an HTTPS connection is used.

### 2.1.2 Flash cookies

Adobe's Flash Player is a plugin for browsers which can be used to show animations and videos. The plugin supports a storage mechanism called *Local Shared Objects* [23]. Normally it is used for example to cache a video or audio file or to save a player's settings chosen by the user. However, all kind of information can be stored there, so it can be used to store unique user identifiers. Earlier on Flash cookies remained untouched when a user deletes its browser data. However, modern versions of Mozilla Firefox, Google Chrome and Microsoft Internet Explorer also delete Flash data when the normal HTTP cookies are cleared [24].

### 2.1.3 Other

Besides Flash there are more plugins that have potentials to store data with, for example Microsoft Silverlight and Java Web App. However, these plugins are becoming more and more rare since the introduction of HTML5. For example, Google Chrome does not support the API that these plugins use any more [21]. Therefore we decided to not elaborate these techniques further here.

## 2.2 Cache Based Tracking Techniques

When a browser accesses a website it is mostly not sufficient to download the corresponding HTML page only. Mostly it does not stop there, because most websites require additional Javascript libraries, images, etc. before it can be fully showed. These files must be downloaded too. At least, if the browser did not download it before. Browsers have built-in mechanisms to prevent that files must be downloaded over and over again. The most used caching techniques are part of the HTTP protocol. Here the *ETag* and/or *Last-Modified* fields are used [7]. Here is saved which version of a file one has and when the file was modified most recently. The particular website determines how these fields are set in its HTTP header, so they can be used to save user identifiers.

Other than storing information in metadata or identifying versions of files, the cached files themselves can be used too [27]. If the web server makes sure that a user only downloads particular files, the combination of files it possesses can be identifiable. For example a website could have some tracking images and each new user gets a unique combination of pictures. During a second visit, all files can be requested and from the information which files a client has and which not the user's identity can be recovered.

## 2.3 Fingerprint Based Tracking Techniques

### 2.3.1 Canvas Fingerprinting

A browser's canvas API included in HTML5 makes it possible to generate images, fonts and drawings locally. The implementation of these functionalities are generally very dependent of implementations made by the operating system and the underlying hardware the browser operates on. For instance, to generate 3D graphics the browser is dependent on the GPU. Each type of hardware generates a slightly different picture. From research of Mowery & Shacham [20] we know that the canvas objects generated are consistent over multiple runs and are sufficiently unique per browser if the object is chosen in a specific way. Then the hash of this image can be used to identify a user. Furthermore, the image can be made invisible for the user, so it is hard to notice. Because the canvas API is key functionality of HTML it cannot be simply disabled. Disabling will lead to losing a lot of functionalities of websites. The only things that still can be done to reduce the uniqueness of images is deliberately adding noise to pictures or make browser implementations less dependent of hardware characteristics to make sure that multiple browsers generate more or less the same picture. A way of making Canvas Fingerprinting more visible is to build a feature into browsers that asks a user's permission if an animation will be made. Then it is suspicious if nothing is visible after granting permission.

At first this technique might look far fetched, but Acar *et al.* [1] showed in 2014 that it was used by some advertising networks at that moment.

Eckersley showed [8] with his browser test *Panopticllick* that in 2010 83.6% of all browsers had an unique fingerprint. Canvas Fingerprinting is one of the methods that has been taken into account in this research. Also other leaked information like the supported fonts or the browser version and its default language contribute to identifying a user. A remark is that not all techniques are equally stable. Some techniques differ in behaviour when tested a few days later [8] .

Overall, we can conclude that fingerprinting techniques can contribute to the identification of users. There are many different techniques available. Those in combination can give a good picture of someone's identity. A remark is that fingerprinting techniques only work on best effort. There is no guarantee that a user can be uniquely identified. Therefore these techniques will mostly be used as a back-up if more accurate techniques fail.

## 2.4 Definitions

### 2.4.1 Super Cookies

Besides the traditional HTTP cookies there are also some more sophisticated methods to identify a specific user. These so called Super Cookies [27] are typically harder to notice for the user and they are usually implemented by different use of certain functionalities in a browser. Another characteristic is that it is much harder or even impossible to delete them. All tracking techniques other than HTTP cookies can be classified in this category. As classification method this definition was therefore not that useful.

### 2.4.2 Evercookies

An evercookie is the idea that multiple tracking techniques can be combined to create a tracking mechanism that is almost unavoidable [1].

With normal HTTP cookies it is easy to track people, but it has the disadvantage for tracking companies that users delete these regularly. With Super Cookies this is harder, but these techniques are more difficult to implement. When methods are combined, tracking is only dependent of techniques that are harder to implement if the easy methods fail. This has the advantage that users can be easily identified by their HTTP cookies and if someone deletes these, the other super cookie techniques can be used to restore them. This is called cookie respawning [1]. This specific method that reconstructs HTTP cookies in particular is also referred to as *zombie cookies*. Evercookies are named after Kamkar's Javascript API Evercookie [14] that can be used to generate them.

These mechanisms have the characteristic that it is almost impossible for users to avoid being tracked without losing functionality, because it uses multiple techniques. If a user disables one recovering technique, the method is still able to identify you using the techniques that are left. The only way to prevent that you are being tracked is to disable practically all techniques that are used.

## 2.5 Related Work about Empirical Analysis

Besides the details of specific tracking methods, it is important to know whether these techniques are used in practice. Several methods are known to find this out. Especially the more automated data collection methods are interesting, because with this information of more websites can be collected efficiently. More websites give a better picture of the situation in practice.

Roesner *et. al.* [22] made a detailed categorization system for tracking services. They distinguish five separate types. An analytics tracker is a party that only follows the user's behaviour within one website. A tracker that follows users across different sites is labelled as vanilla. On top of these two types, they also distinguish parties that need a redirect or pop-up, parties that get identifiers leaked from other sites and trackers that are visited directly by the user. To identify which tracking service belongs to which category they need to distinguish user identifiers from other storage. For this they used a combination of characteristics how a user identifier would look like (size, etc.) and whether the identifier remains equal over more visits to the same website.

In research of Metwalley *et. al.* [18] user tracking was recognized by checking for possible user identifiers in URLs. Here the method of recognizing identifiers is explained more deeply than in the research of Roesner *et. al.* [22]. Websites were visited by multiple simulated users multiple times. If a identifier was the same all times the website was visited and it changed when another user visited it, it is labelled as an user identifier. This method can be easily extended to user identifiers stored in other ways than in URLs.

Research of Krishnamurthy & Wills [16], also used by Falahrastegar & Mortier [9], did not include any check for user identifiers at all. A distinction between first parties and third parties is made and there is no automatic classification about which kind of third party it was.

## Chapter 3

# Tracking Techniques in HTML5

In 2014 W3C published the official HTML5 recommendation [13]. This new specification was made to improve the support for interactive web applications [13, section 1.1]. The new standard also introduced some new techniques that can be used to track users. In this chapter we focus on storage functionalities within the HTML5 specification: Local Storage (Section 3.1) and IndexedDB (Section 3.2). We tell how they work and what the tracking potentials are. In Section 3.3 we explain how tracking using these techniques works. We did not examine fingerprinting and cache based tracking techniques included in HTML5.

HTML (Hypertext Markup Language) [13] is the language that is used to specify the content and the layout of websites. Web browsers are then able to render the website and show it to the user as it was intended to look like. HTML is normally used on top of HTTP. Features that are not supported by the chosen HTML version can be added using plugins. Adobe Flash and Microsoft Silverlight are examples of such plugins. We visualized this idea in Figure 3.1. We put browser plugins on top of the stack. Strictly seen communication of browser plugins is completely separated from the web protocol stack, because the client-server communication of plugins is not included in HTML. Yet, we did this to visualize that generally browser plugins build on HTML. Plugins are only needed when the particular functionality is missing in the underlying structure.

With HTTP cookies people could only be identified on moments when they contacted the corresponding server. In HTML5 this is not a restriction any more, because these elements are available via JavaScript, so offline scripts can also track user behaviour in between these moments. This information can then be sent to the tracking party afterwards. Previously this was only possible with scripts that run on a page and immediately send the information they collected to a tracking party. This has more overhead for

<i>Browser plugins (i.e. Adobe Flash)</i>
HTML
HTTP/HTTPS
TCP
⋮

Figure 3.1: Most common web protocol stack

the tracking party. More real-time user tracking gives tracking parties the opportunity to adapt their website on the fly (i.e. change advertisements) to what a user does.

Considered the protocol stack visualized above, HTTP cookies are always present as an alternative for HTML5 storage techniques. This is because HTTP is needed as underlying protocol for HTML. There is no intention too to change this due to backwards compatibility in the new version of HTTP (HTTP/2) [6].

### 3.1 Local Storage

The most well known HTML5 storage technique is Local Storage which is included in the Web Storage API [12]. This API provides a similar technique as HTTP cookies. The Storage objects are implemented as key/value pairs. Websites can use several objects to store the information they need and later on the information can be retrieved by requesting the corresponding identifying key. The functionality that the storage interface provides is similar to HTTP cookies. Because HTML is build upon HTTP developers usually have both techniques available all the time.

There are two implementations of the Storage interface that can be used:

**Session Storage** This technique only stores information for one browser session only. The idea of this is similar to the idea of non-persistent HTTP cookies. The only difference is that session storage is really limited to the context of one particular session. Non-persistent HTTP cookies on the contrary are shared between multiple tabs of the same website and thus not limited to one session.

For tracking purposes Session Storage can only be used to track a user within one website during one specific session. Therefore this particular technique is not that useful as a stand-alone tracking technique. Because of this we from now on only pay attention to Local Storage.

**Local Storage** Local Storage is the implementation used for persistent data storage. There are some important differences between HTTP cookies

and Local Storage. At first Local Storage objects do not know an expiry date. So, in principle, Local Storage objects are stored forever, unless they are deleted explicitly at some time. Secondly, HTTP cookies are sent to a web server automatically in all requests a web browser does. In Local Storage resources must be requested manually via JavaScript. For a web server to get these resources they must also be included in a request to the server manually. This requires more overhead in JavaScript code.

### 3.1.1 Functioning

Storing information in Local Storage is rather easy. With some straightforward JavaScript statements information can be permanently stored in the browser.

```
// Storing information with key "test"
localStorage.test = "Hello world!";
// Retrieving information
var x = localStorage.test;
```

The information that is stored in Local Storage can be viewed in the development kits of most modern browsers (i.e. Google Chrome, Mozilla Firefox and Microsoft Edge). An example of this view in Google Chrome can be seen in figure 3.2.

## 3.2 IndexedDB

For storing larger amounts of data Local and Session Storage are not that practical. Per key only one value can be stored. Web developers should then need to pay attention to ways to structure their data, let alone the computational overhead this structuring would involve. To make this easier for them another data storage method has been added to HTML5: IndexedDB [17]. This API provides a way to save JSON strings in a structured way. Furthermore it makes it possible to set indexes to specific items in the database to find them quickly. The system does not behave like a traditional relational database. It is more like NoSQL-techniques [15]. Obviously this storage technique stores data persistently.

A SQL based database system does also exist: Web SQL [10]. However, because of a lack of independent implementations of this technique W3C decided to not further maintain this standard. Therefore we ignore this technique in this research.

### 3.2.1 Functioning

Storing data in an IndexedDB database is more complex than it is in Local Storage. At first a connection must be set up to one of the local databases

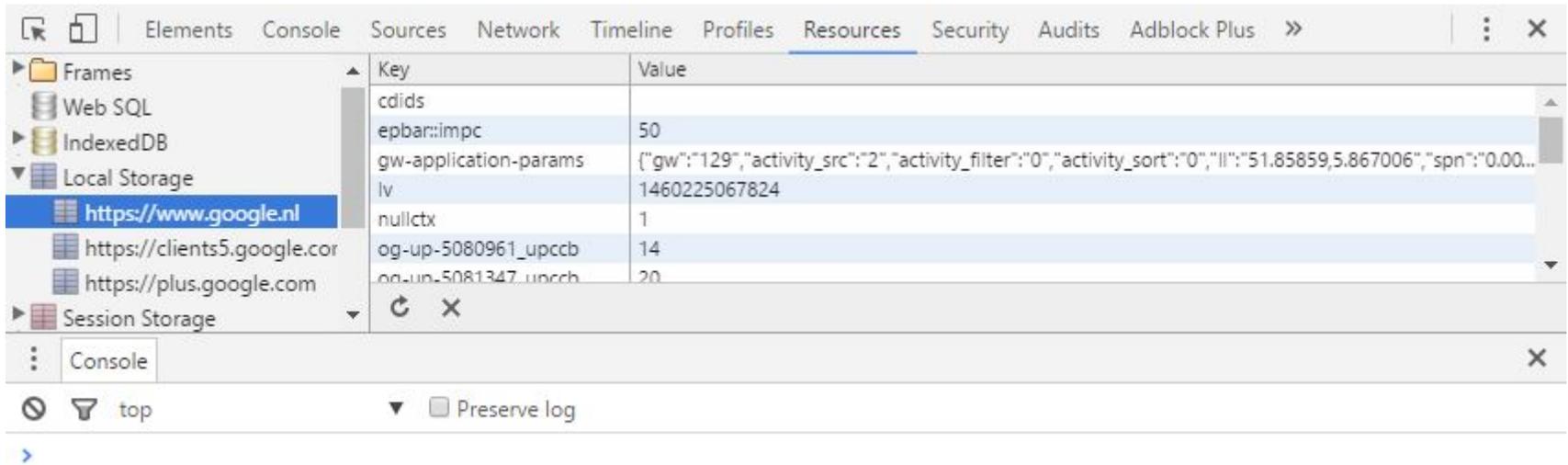


Figure 3.2: Local Storage resources of `google.nl` displayed in Google Chrome

stored by the browser. If the database does not exist yet, browsers will make one.

When a connection has been made objects can be stored using database transactions. With transactions objects from the database can be requested and then items can be added to those objects. An item must be specified as JSON string.

IndexedDB uses error handling as default behaviour. Therefore most methods do not return their values immediately. At first event handlers must be specified what to do if an operation succeeds or when it gives an error.

Reading from and writing to IndexedDB databases can only be done via JavaScript. A JavaScript example how to add and remove a simple database can be found in Appendix A.

As well as with LocalStorage, IndexedDB databases can be displayed in the development kits of Google Chrome and Mozilla Firefox. An example of this view can be seen in Figure 3.3.

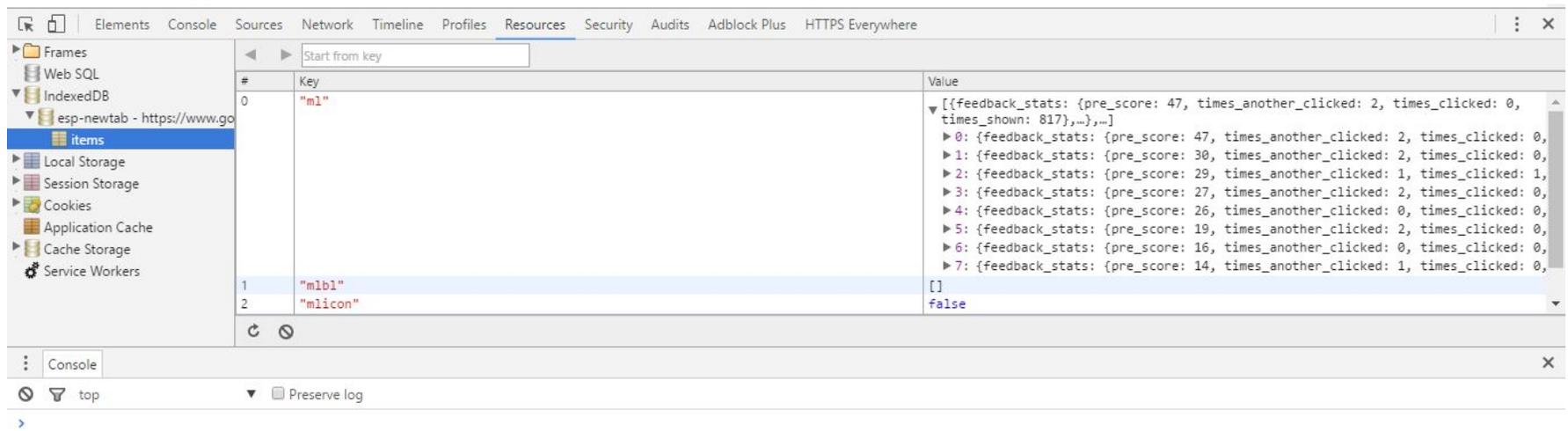


Figure 3.3: IndexedDB resources of google.nl displayed in Google Chrome

### 3.3 Tracking Method

In Chapter 4 we discuss the privacy measures taken in browsers to prevent user tracking via Local Storage and IndexedDB. Because of these measures it is not trivial how user tracking can be done.

From the implementation details of privacy measure 1 described in Section 4.2 we can conclude that Local Storage and IndexedDB resources can only be accessed by the exact origin that stored the information. So information stored by one website cannot be read by another.

The appliance of Same Origin Policy makes that if tracking services want to track a user across different websites, they must embed a frame on all websites. If the frame uses the origin of the particular tracking service, its Local Storage and IndexedDB resources can be accessed from within this frame.

A tracking service usually also wants to know the details of the website the user visits. Therefore the embedded frame needs to communicate with the parent website to get these details. The details can then be linked to a user identifier by the frame and subsequently all data can for example be forwarded to the tracking service.

Communication between frames can potentially be done in several ways:

- The most simple one is that the details of the website are mentioned in the source URL of the frame. For example GET parameters can be used.
- It might be possible for frames to directly access the DOM of parent frames. In this way the details of the website can simple be requested via JavaScript.
- There is also a special communication system available in browsers to communicate between frames: Web Messaging [11]. Calling the JavaScript function `postMessage()` of another frame makes it possible to send messages across frames. The receiving frame must explicitly listen for messages, so co-operation of both the sending and the receiving side is needed. According to the specification [11] there are no built-in restrictions about this communication, so with this the Same Origin Policy can be circumvented if both parties work along.

Not all options for cross frame communication can be used in practice due to protection measures in web browsers. A browser comparison about what is and what is not possible as regards to cross frame communication can be read in Section 4.3.

The general idea of making user tracking possible via Local Storage and IndexedDB using embedded frames is visualized in Figure 3.4.

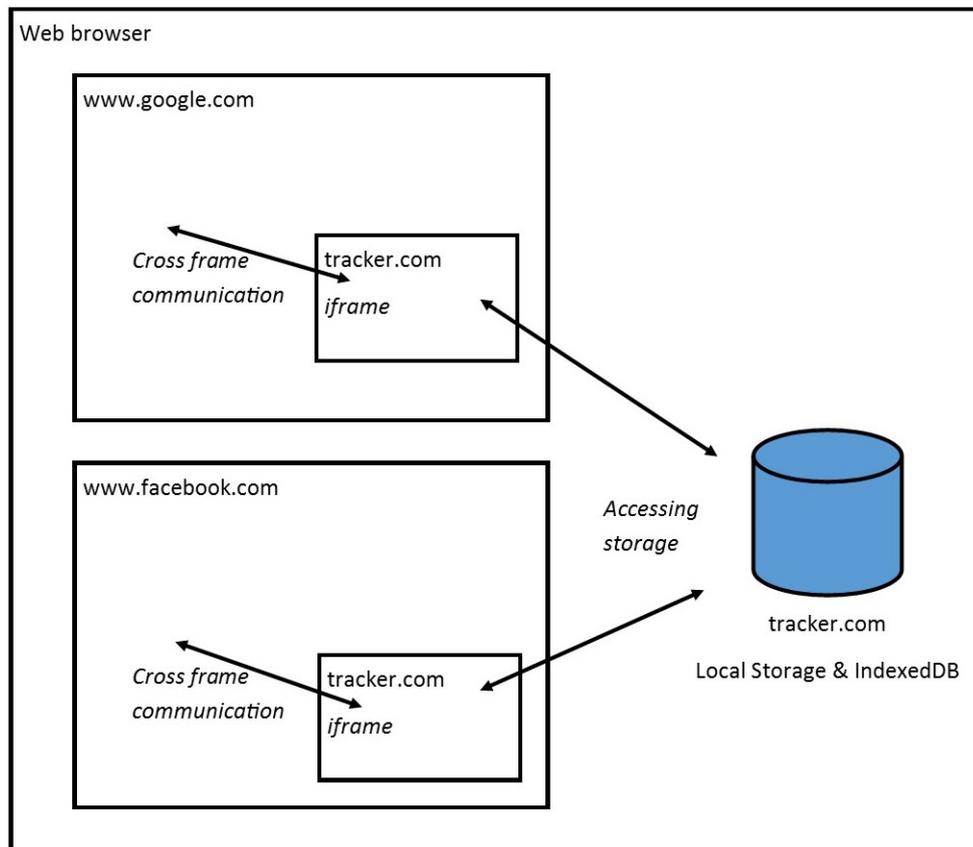


Figure 3.4: Graphical representation of user tracking using Local Storage and/or IndexedDB

## Chapter 4

# Privacy Measures in HTML5

In the specifications of HTML5 storage techniques [13] [17] recommendations are mentioned about the prevention of user tracking. These are discussed in Section 4.1. Because these recommendations are rather non-committal we decided to do some research in how these recommendations are implemented in web browsers. This is discussed in Section 4.2. From Section 3.3 we know that cross frame communication is needed to enable third party user tracking via Local Storage and IndexedDB. The privacy measures taken to prevent cross frame communication are discussed in Section 4.3.

### 4.1 Tracking Prevention Recommendations W3C

Privacy issues are mentioned at the end of both the specifications of Web Storage [12, Section 6] and IndexedDB [17, Section 4]. Because the content of both sections is more or less the same, we will discuss both together.

In both sections the possibility that features of the techniques can be used for user tracking is acknowledged. The most important recommendation is that the storage techniques should be treated the same as HTTP cookies. What this exactly involves is described at privacy measure 1 in Section 4.2. Furthermore there are no compulsory prevention methods.

The specifications do mention some measures that browsers can use to reduce the risk of user tracking:

- Blocking third-party storage completely
- Let stored data expire after some time
- Let users white-list websites to use local storage
- Use blacklisting to prevent that known tracking domains use it
- Make domains that store data visible for users

What stands out is that the advised measures are rather the same to the measures taken to prevent user tracking with HTTP cookies.

Because HTML has a higher level of abstraction than HTTP, not all prevention methods against tracking with HTTP cookies can be applied to Local Storage and IndexedDB too. For example something like HTTP-only cookies can not be constructed in Local Storage and IndexedDB, because HTML cannot interfere in the underlying HTTP protocol. Furthermore all measures related to expiry dates cannot be applied, because Local Storage and IndexedDB do not know such. Therefore the only option that is mentioned is to set a fixed expiry date for data.

## 4.2 Comparison: Tracking Prevention in Browsers

We decided to compare four browsers on their performance on each privacy measure. We chose the most recent versions of Google Chrome, Mozilla Firefox and the default browsers in Windows (Microsoft Edge for Windows 10 and Microsoft Internet Explorer 11 for earlier versions). We tested the behaviour just in normal conditions and we did not try to find minor implementation flaws.

At first we describe the general idea of the testing set-up we used in Section 4.2.1. Subsequently we discuss all tests we have done in Section 4.2.2.

### 4.2.1 Testing Set-up

To test the different measures we made some tracking examples using iframes to test the behaviour of browsers. To be able to test these we made a set-up with two virtual machines assigned to two different local URLs. One virtual machine runs the website that is being tracked (from now called tracked party) and the other runs the website of the tracking service. The website of the tracked party includes an iframe to the website of the tracking service. A visualization of this set-up can be seen in Figure 4.1. We let both the website of the tracked party and the website of the tracking service use Local Storage and IndexedDB to be able to make a distinction between first party and third party usage.

A limitation to this set-up is that Microsoft Edge does not support the possibility to open websites hosted by a local virtual machine. Therefore we tested this browser by injecting JavaScript code manually in a normal website that contains iframes. JavaScript code can be injected via the JavaScript console included in the browser's developers tools. Here the frame in which the code must be executed can also be chosen. Testing by executing code manually has the risk that there could be differences in behaviour between injecting code manually and how code would be executed normally. For

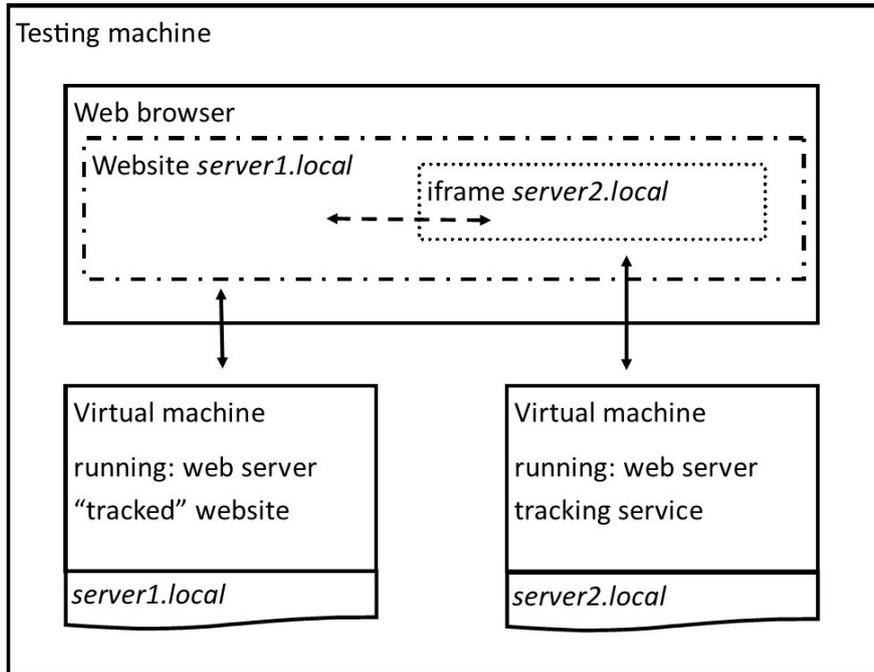


Figure 4.1: Visualization testing set-up using virtual machines

example security and privacy measures might be less strict for manual execution.

#### 4.2.2 Tests

In the table on the next page an overview is given of all results we found. A detailed description of the particular result is given afterwards in the section that corresponds with the number in the table. The privacy measures correspond to the measures mentioned in W3C specifications (Section 4.1). Privacy measure 7 is added, because while testing privacy measure 3 we discovered that some browsers applied this.

An explanation of the symbols used in the table is given below:

- ✓ Implemented and enabled by default
- Feature is implemented, but is optional (not enabled by default)
- \* Partially implemented
- ✗ Not implemented
- # Feature is not supported by browser

Privacy measures		Local Storage				IndexedDB			
		Chrome	Firefox	Edge	IE	Chrome	Firefox	Edge	IE
1.	Implemented just like HTTP cookies								
1a.	Same Origin Policy								
1a.i.	Host	✓	✓	✓	✓	✓	✓	✓	✓
1a.ii.	Protocol used	✓	✓	✓	✓	✓	✓	✓	✓
1a.iii.	Port used	✓	✓	✗	✗	✓	✓	✗	✗
1b.	Data deleted when cookies are deleted	✓	✓	✓	✓	✓	✓	✓	✓
2.	Blocking third party storage	O	O	✗	✗	O	O	✗	✗
3.	Let data expire after some time								
3a.	Normal browsing mode	✗	✗	✗	✗	✗	✗	✗	✗
3b.	Private browsing mode	✓	✓	✓	✓	✓	#	#	#
4.	White-listing	O	O	✗	✗	O	O	✗	✗
5.	Blacklisting								
5a.	Block certain domains	✓	✓	✗	✗	✓	✓	✗	✗
5b.	Export feature to share blacklists	*	*	*	✗	*	*	*	✗
6.	Make domains that store data visible for users	✓	✓	✓	*	✓	✓	*	*
7.	Data not shared between multiple tabs in private browsing mode	✗	✗	✓	✓	✗	#	#	#

1. *Implemented just like HTTP cookies*

Because both techniques should be implemented like HTTP cookies, Same Origin Policy should have been implemented for Local Storage and IndexedDB too. Furthermore all Local Storage and IndexedDB databases should be deleted when the cookies are deleted by a user.

**Method** Same Origin Policy distinguishes three elements that specify a different origin: the host name of a website, the protocol it uses (i.e. HTTP/HTTPS) and the port number the service runs on [28]. The first two elements (labelled with 1a.i. and 1a.ii. in the table) can be tested relatively easy. We can just try to access the Local Storage and IndexedDB resources of a website with another host name to test measure 1a.i. and of two pages of the same website where another protocol is used to test measure 1a.ii. Testing whether browsers distinguish different ports for an origin is harder to test, because it is harder to find websites that serve websites on multiple ports. Therefore we hosted a local website by ourselves to test this. We configured it to serve the same website on two ports. As a result we could just test measure 1a.iii in the same way as we tested the other elements of the Same Origin Policy.

Testing whether Local Storage and IndexedDB resources are deleted when other browser data, like HTTP cookies, are deleted, can be done by saving some data with both mechanisms and then delete the browser data. Afterwards we can test whether the added data is still available or not.

## Results

- a. All four browsers implemented the Same Origin Policy. The only remark is that Internet Explorer and Microsoft Edge make no difference between URL with different port numbers. For example the URLs `http://server1.local/` and `http://server1.local:81/` belong to the same origin in Internet Explorer and Microsoft Edge. All four browsers only grant access to one Local Storage and IndexedDB domain per same origin. It is not possible to change origin with JavaScript except for executing it within iframes. The implementation of the Same Origin Policy is more strict than for HTTP cookies, because sites can also read cookies for less specific domains (`www.google.com` can read HTTP cookies of `google.com`). For Local Storage and IndexedDB this is not possible. Access is restricted to the domain itself.
- b. All four browsers also deleted the stored data in Local Storage and IndexedDB when the cookies were deleted (privacy measure 1b). In

Internet Explorer Local Storage resources are maintained in tabs opened before the data was deleted.

2. *Blocking third party storage*

Most browsers have options to block third party access or cookies. In here we examine whether these option also involve Local Storage and IndexedDB.

**Method** From the results of measure 1 we know that the only way a third party could access its resources is when it is included in an iframe. If third party access is disabled, the tracking service should not be able to access its resources in Local Storage and IndexedDB from within the iframe. Therefore we let the tracking service try to save and read some information and then we can verify whether it is allowed or not. To test this we use the testing set-up described in Section 4.2.1.

**Results** All browsers have an optional functionality to block third party cookies. In Google Chrome and Mozilla Firefox this option also blocks third party access for Local Storage and IndexedDB. Internet Explorer does not do this. We found by using manual tests that Microsoft Edge reacted the same as Internet Explorer. However, we have to take into account here that manual tests might not give a right impression.

3. *Let data expire after some time*

Expiry dates do not exist in the specification of both Local Storage and IndexedDB. There is also no indication in browsers that they implemented expiry dates anyway. There is a possibility that a browser implemented it as a hidden feature, for example that resources are deleted after a fixed time. However, if such feature would have been implemented it would be logical that browsers documented this clearly, because for web developers it is useful to know that this behaviour exists. Therefore it is not likely that browsers implemented expiry dates.

All four browser do have a private browsing mode. In this mode browsers delete all stored data after the browser is closed. This can be seen as an expiry system.

**Method** To test the behaviour of browsers in private browsing mode on Local Storage and IndexedDB we used the testing set-up described in Section 4.2.1. We opened the website in private browsing mode. After that we first opened the same site in an other browser tab to

test whether information is isolated within one browser tab or not. After that we restarted the browser completely to test whether all Local Storage and IndexedDB data is deleted then. To verify whether the data is deleted, we reopen the website and check whether the data is still available.

**Results** As we already described in the method, expiry dates are not taken into account in the specification of Local Storage and IndexedDB.

In private browsing mode we found that all browsers delete the Local Storage resources after the browser is closed. In the case of IndexedDB Google Chrome reacts the same in this as for Local Storage. All the other browsers we looked into chose to not allow that IndexedDB is used in private browsing mode. Applications that rely on IndexedDB will therefore not work in private browsing mode in these browsers.

We also found some implementation differences of the private browsing mode between browsers. These differences are described at measure 7.

#### 4-5. *White-listing and blacklisting*

Blacklisting and white-listing functionality are combined in all browsers we tested. Therefore we also combine the test of these two features.

**Method** To verify whether the blacklisting and white-listing functionality of a browser work we can again use the testing set-up described in Section 4.2.1. If we blacklist or white-list the testing website to use cookies and other storage mechanisms, we can check with that website whether these functionalities work or not at that moment.

No browser has the feature implemented to share blacklists or white-lists. Users can only sync their browsers on multiple devices to maintain the same settings. Internet Explorer does not have such a feature at all. Because of this it is not necessary to construct a test for this. No browser can be configured in such a way, so all sites will fail this test.

**Results** In Google Chrome and Mozilla Firefox it is only possible to blacklist or white-list a domain completely. You cannot block specific information stored by for example google.com that is used for tracking. Then you also have to block storage requests that are necessary to use the site properly. It is not possible to accept usage when it occurs. In this way you could distinguish tracking from normal usage. Now all settings must be filled in before hand. Microsoft Edge and Internet Explorer do have a feature to blacklist and white-list, but this does

not work for Local Storage and IndexedDB usage. It is only meant for HTTP cookies.

6. *Make domains that store data visible for users*

The origin of storage items is saved by all browsers to achieve Same Origin Policy. Internet Explorer only has no way to show this list to the user. All other browsers we tested have such a feature included in their development kit.

7. *Data not shared between multiple tabs in private browsing mode*

When examining measure 3 we noticed that there is a difference in implementation of private browsing mode between Google Chrome and Mozilla Firefox on one side and Microsoft Internet Explorer and Microsoft Edge on the other side. Mozilla Firefox and Google Chrome share data between different private browsing tabs. All data is erased when the browser is closed. In Microsoft Internet Explorer and Microsoft Edge all private browser tabs are isolated from each other.

### 4.3 Comparison: Protection Against Cross Frame Communication in Browsers

In Section 3.3 we mentioned three straight forward ways to exchange information between frames on a website. Web browsers also realized that cross frame communication, especially between frames with a different origin, involves privacy issues. It could be that certain parties learn information from another frame without the user or even the frame that contains the information wanting so. Therefore protection measures are taken to prevent this by web browsers. Frames cannot send information directly to other parties over the internet, because for sending information via JavaScript over the internet Same Origin Policy is also applied [26, Section 6]. This makes that cross frame communication is necessary to send information to third party tracking services.

#### 4.3.1 Parameters in Source URL

Passing information using source URLs of frames is the least harmful way of leaking information to other frames. The initiative of making information available always rests with the frame that possesses the information. It is hard for browsers to check whether the source URL of frames embedded in a website contains sensitive information. Checking URLs would limit the functionality of frames and this could potentially break websites that use frames. Therefore there is no indication that web browsers do this.

Using the source URL field of frames communication can be done only in one direction. The outer frame can set the initial source URL of frames.

All information put in this URL can be seen by the embedded frame and by the server of the tracking service that handles the request to load the frame. All changes to the source URL that the embedded frame makes, for example going to another page, cannot be seen by the outer frame. Thus, the embedded frame cannot use the source URL to send information to an outer frame.

For tracking the most important direction is the communication from the outer frame to an embedded frame to communicate information about the website opened in the outer frame. This way of doing cross frame communication makes this possible.

### 4.3.2 Cross Frame DOM Access

To check whether browsers have protection measures against cross frame DOM access we again used our testing set-up described in Section 4.2.1. From the embedded frame we tried to access the DOM of the most outer frame by testing we could execute the JavaScript statement `top.document.URL`. Again we tested whether this worked in Google Chrome, Mozilla Firefox, Microsoft Edge and Microsoft Internet Explorer.

From this test we can conclude that all four browsers also apply Same Origin Policy on JavaScript DOM access, which means that in none of the browsers it was possible to access data of another frame. The JavaScript DOM can therefore not be used to do cross frame communication. A note is that the test set-up does not work for Microsoft Edge. In this browser we did the test manually using the JavaScript console.

### 4.3.3 postMessage API

Finally we also tested whether the postMessage API can be used to do cross frame communication. The API is designed for this purpose, so it is expected to be possible. However, browsers may have applied some kind of Same Origin Policy on this communication.

For this test we also used the testing set-up described in Section 4.2.1. We sent messages using the postMessage API from the tracking frame to the outer website and vice versa. Again we tested the behaviour of the web browsers Google Chrome, Mozilla Firefox, Microsoft Edge and Microsoft Internet Explorer.

The tests proved that the four tested web browsers allow cross frame communication in both directions. By default Same Origin Policy is not applied to this. However, receivers of messages must explicitly define listeners to receive the messages. A postMessage listener receives all messages that are sent to that frame. Checking origins of messages can be done manually. Thus, this means that for tracking services this privacy measure is easily circumventable.

## Chapter 5

# Empirical Analysis of HTML5 Tracking Techniques

In Chapter 3 we explained the two storage techniques available in HTML5 that can be used for user tracking: Local Storage and IndexedDB. The possibility to track with these techniques does not mean that it is also used for this purpose. In this chapter we examine whether this is the case. At first we do a manual analysis of some websites. This is described in Section 5.1. After that we use a browser extension in Google Chrome to check for tracking usage of Local Storage and IndexedDB on well known websites more systematically. The method is discussed in Section 5.2 and the testing results in Section 5.3.

### 5.1 Orientation phase

To get an impression of how tracking techniques are used by advertising companies we decided to first check some websites manually. We will use this impression to implement a more systematic method to find which parties use the HTML5 storage techniques. This more systematic method is described in Section 5.2.

For our manual analysis we based ourselves on the website of Google ([www.google.com](http://www.google.com)) and the websites of two large newspapers in the Netherlands: De Telegraaf ([www.telegraaf.nl](http://www.telegraaf.nl)) and AD ([www.ad.nl](http://www.ad.nl)). We chose these because the websites of both newspapers use a high number of tracking and advertising companies. We added Google, because we know that this site uses an IndexedDB database what seems to be for tracking purposes (see Section 3.2).

Our first impression is that both techniques are not used that much. At the website of the Telegraaf two out of nine third party trackers that use embedded frames use Local Storage and none of those use IndexedDB. From the two trackers that use embedded frames only `c.betrad.com` saved



be given whether users really face these kind of techniques. A disadvantage is that sometimes it is hard to determine whether something is a tracking technique or just a normal feature that the website provides. For example identification strings in the storage of websites can also be necessary for authentication and not directly for user tracking. This can be solved by checking multiple times. User identifiers will not change over time. However, most other variables will also be relatively fixed, especially in the short term. Hence, for quick evaluations this does not solve the problem. This would only work if Local Storage and IndexedDB usage is analysed for a longer time.

- *Check for known tracking services which techniques they use*  
This approach takes away the difficulty that elements must be identified as being used for tracking. If elements originate from an actual tracking company it is likely that this is meant for tracking. As a testing sample a list of large tracking companies can be used. A drawback of this approach is that we miss new techniques that smaller companies might use. In this approach the assumption lies that there is a high probability that if techniques are used, then they are also used by the major tracking parties.

Because we want to focus on whether and how techniques are used instead of which websites exactly do this, we chose the second approach. To use this method we have to define which tracking parties we will take into account and which websites we are going to analyse to test the usage. This is discussed respectively in Section 5.2.1 and Section 5.2.2. Finally we described the functioning of the Google Chrome extension which we made to do the analysis in Section 5.2.3.

### 5.2.1 Chosen Tracking Parties

Research of Metwalley [19] in 2016 provides a list of the 20 most pervasive online tracking services found by tests with actual users:

1. Doubleclick
2. Google Analytics
3. Google Syndication
4. Google Ad Services
5. Scorecard Research
6. IMR Worldwide
7. ADNXS
8. Criteo
9. Rubicon Project
10. Serving Sys
11. Adform
12. Google Tag Services
13. Google Tag Manager
14. Pubmatic

- |               |                |
|---------------|----------------|
| 15. Ads.yahoo | 18. Bluekai    |
| 16. Addthis   | 19. Mathtag    |
| 17. Turn      | 20. Ad Advisor |

We will analyse the usage of HTML5 storage techniques of the services mentioned in the list. Because we know these are tracking services, we know that if we identify usage, this usage will contribute to tracking activities.

### 5.2.2 Chosen Websites to Analyse

To be able to observe which storage techniques the tracking services use for user tracking, we need to specify a test set of websites to analyse on. Because we already know from Section 5.2.1 that the chosen list of tracking parties involves the most pervasive ones, the most important criterion for a website is that it is likely that one of the chosen tracking services from Section 5.2.1 is tracking users on that website.

The test set we are going to use is composed by ourselves by making a manual selection of Alexa Top 500 [2] and the Alexa top sites in the Netherlands [3]. We chose not to include websites that only allow elaborate access when a user is logged in, because then users can also be identified by their login credentials. Another selection criterion was to include mainly news and entertainment websites. Most of these websites have business models based on advertising, so the presence of tracking services there would be logical. The complete list can be found in Appendix B.

### 5.2.3 Browser Extension for Collecting Results

To find out how much the HTML5 storage techniques Local Storage and IndexedDB are used by the 20 tracking services mentioned in Section 5.2.1 we visit all websites chosen in Section 5.2.2. From measure 1 of the browser comparison in Section 4.2 we know that third party storage is only possible if that party is embedded within a frame of the website that wants to track its users. To prevent that we have to check the stored resources of all frames on a website manually we constructed a browser extension for Google Chrome that collects that information of all tracking services listed in Section 5.2.1.

At first we will clarify the terminology used in this section and Section 5.3. A *tracking service* (i.e. Doubleclick) can use multiple *domains*. In most cases this will be different subdomains of the tracking service's main domain. For example `x.doubleclick.com` and `y.doubleclick.com` are two domains, but belong both to one tracking service. A *website* is a combination of different *pages*. For example the website `www.ru.nl` can have pages like `www.ru.nl/page1.html` and `www.ru.nl/page2.html`.

The extension tests for every frame whether its originating domain is related to one of the parties listed in Section 5.2.1. If so, it checks whether Local Storage, IndexedDB or both are used in that particular frame. Because we know it involves a tracking company it is plausible that this usage is meant for tracking. However, we do not determine whether the information stored can be used as a unique identifier. We only measure usage in general. The extension can only handle one active browser tab and to get all results correctly it is important to wait until all resources of a page are fully loaded.

### Detailed explanation browser extension

The browser extension mainly consists of two scripts:

- **check\_current\_frame.js**  
The browser extension is configured to execute this script in every frame a page contains. The script checks whether the originating domain of the particular frame is related to one of the 20 tracking services listed in Section 5.2.1. If this is the case the script checks whether that domain stored Local Storage and/or IndexedDB resources and sends this information to the background script.
- **background.js**  
A problem that we had to solve was that Same Origin Policy does not allow cross origin DOM access in JavaScript and this policy is maintained for Chrome extensions. This means that we could not find out the URL of the top page when the content script `check_current_frame.js` is executed in an embedded frame of that page. The solution we came up with is to let all frames communicate the information they found to a background script. Background scripts belong to the browser extension and therefore have more permissions. They can easily request the URL of the active tab at that moment. Therefore in the background script listeners are implemented that are executed when the URL is changed in the active browser.

The other listener that is implemented receives the messages from all the `check_current_frame.js` scripts that are running. It first checks whether the tracker was already found earlier. If so, the information about that tracker is updated. A new website where a tracker was found that was already known is added to the list of websites. When the website was already known too, only the information about the usage of storage techniques is updated. The information is only updated when nothing was found in the first place and after a rerun Local Storage and/or IndexedDB usage is recorded. We did this to prevent that we loose information when there are differences in usage between

multiple visits to a page. If the tracker is still unknown, all data the content script collected is stored.

The data is stored in the `saved_trackers` object of the background script. To make it possible to view the results we implemented a popup (`popup.html` and `popup.js`) to view the information found in a table. After a lot of browsing the popup in the browser is too small to see all information. Therefore we added a feature to open the result table in a separate browser tab.

The program code of this extension can be found in Appendix C.

## 5.3 Results

Given the test set of webpages described in Section 5.2.2 we found frames of 13 out of the 20 tracking services listed in Section 5.2.1. The other 7 tracking services were not present at one of the websites in the test set using a frame. This does not necessarily mean that those parties do not track on one of these sites. Not for all tracking techniques an embedded frame is needed. For example HTTP cookies are already sent when a server request is done to the particular domain. The full table with results can be found in Appendix D which can only be found in the digital version of this document due to the size.

In total we found 40 different tracking domains to be active on at least one of the tested pages. This is caused by the fact that some tracking services use multiple domains. A summary of the results we found is showed in Table 5.1. In some cases there are differences in usage between different domains of one tracking service. In these cases the result is marked with (some).

### 5.3.1 Local Storage Usage

Our browser extension identified a frame belonging to one of the 20 tracking services on 89 pages we visited during the test. All together those 20 tracking services used 40 different domains. Out of the 40 domains found, 6 use Local Storage resources. It concerns all four domains of IMR worldwide, one domain of ADNXS and one of Rubicon Project. This means that only 15% of all tracking services we have looked at use Local Storage.

The tracking domain `ib.adnxs.com` has a striking result concerning the usage of Local Storage. The tracker was found at four different pages, but only at one page Local Storage usage occurred.

The opposite situation also occurred at the tracking domain `seccdn-gl.imrworldwide.com`. From the three pages we visited where this tracking domain is active, Local Storage was used by this tracking domain on two pages. All these three pages were part of the website `https://www.theguardian.com`. The two pages where the usage was found contained a news article.

Table 5.1: Summarized results empirical analysis

Tracking service	Number of different domains found	LocalStorage used	IndexedDB used
DoubleClick	14	No	No
Addthis	1	No	No
Criteo	3	No	No
Bluekai	2	No	No
IMR Worldwide	4	Yes	No
Google Syndication	1	No	No
ADNXS	5	Yes (some)	No
Rubicon Project	2	Yes (some)	No
Turn	2	No	No
Adform	1	No	No
Mathtag	1	No	No
Pubmatic	3	No	No
Serving Sys	1	No	No

The other one was a navigation page. Because the navigation page was loaded first, it can be that the data is not stored initially.

Thus, apparently there are differences in usage between the same tracker on different websites or even different pages on the same website.

### 5.3.2 IndexedDB Usage

None of the websites we looked into uses IndexedDB storage. No local databases could be found belonging to one of the 20 tracking services. Because IndexedDB databases as storage for tracking is more complicated than for example Local Storage it could be that this method is only used in special circumstances, for example on more complex webpages. A limitation in this result could therefore be that we did not run into one of those pages in the tests. Nevertheless, we can conclude that in normal circumstances it is not used.

## Chapter 6

# Future Work

In this research we looked into the tracking techniques in HTML5 that use storage functionality to recognize users. From Chapter 2 we know that there are also fingerprint and cache based tracking technique. This means that there is still an open field for future research. We also have some future work continuing our work.

### Open Field HTML5 Tracking Techniques

In the category fingerprint based tracking techniques in HTML5 a lot is already known about canvas fingerprinting. Potentially there are more techniques constructable in this category than canvas fingerprinting only.

We also found out that HTML5 has a cache functionality built-in: manifests [13, section 5.7]. Open for future work is whether it is possible to construct a cache based tracking technique with manifests and if so: what are the pros and cons.

### Empirical Analysis

In Section 5.2 we chose to focus on only analysing the usage of Local Storage and IndexedDB of well known tracking companies. For future work a method can be created that makes it possible to also analyse without a fixed list of trackers. In here it is needed to find a way to determine whether resources in storage can be used to identify users or not. Without limitations about tracking parties the usage of HTML5 tracking techniques can be analysed for any website. With this it is for example possible to analyse websites with sensitive content (i.e. medical advice) on usage of HTML5 tracking techniques.

## Countermeasures

Besides for countermeasures against tracking using Local Storage and IndexedDB built-in in browsers, there are also privacy tools available that tend to block tracking techniques. Tools you can think of in here are for example Adblock, Ghostery and BetterPrivacy. We do not know yet whether these tools also block tracking when Local Storage or IndexedDB are used.

## Tracking in Web Applications

Because Local Storage and IndexedDB can be accessed using JavaScript it is possible to store information about the usage of an application without communication with the server is needed. This offers potentials for more advanced tracking methods. For example it is possible to analyse what the exact user interactions are. A proof of concept could be made to show that this kind of tracking is possible.

## Legal Consequences

According to Dutch telecommunication law<sup>1</sup> websites that operate within the Netherlands must ask a users permission before information is stored at someone's device. This suggests that websites must also ask permission when Local Storage or IndexedDB resources are used for tracking purposes. This claim could be sorted out more deeply. It is also interesting whether privacy guidelines give recommendations about this. On top of that attention can be paid to the situation in other countries.

## Sandboxing

In the HTML5 specification [13, section 4.7.2] an optional sandboxing feature is introduced to make it possible to limit the possibilities of third parties embedded within iframes. Open for future work is how this feature could be used to prevent user tracking.

---

<sup>1</sup>[http://wetten.overheid.nl/BWBR0009950/2016-01-01#Hoofdstuk11\\_Paragraaf11.1\\_Artikel11.7a](http://wetten.overheid.nl/BWBR0009950/2016-01-01#Hoofdstuk11_Paragraaf11.1_Artikel11.7a)

## Chapter 7

# Conclusions

Since the HTML5 specification has been developed at a time in which user tracking was already known as a threat, one would expect that privacy considerations are important for new storage techniques. Taken into account the specifications of the storage techniques Local Storage and IndexedDB, attention is paid to privacy aspects. However, most of these recommendations only offer options (see Section 4.1) to reduce the effects of user tracking. These options are not mandatory and therefore mainly dependent of the willingness of browser manufacturers. The only more strict recommendation is that the new storage techniques Local Storage and IndexedDB must be treated in the same way as HTTP cookies.

### Privacy Measures

All browsers we examined in Section 4.2 have similar privacy measures for Local Storage and IndexedDB. Same Origin Policy is even implemented more strictly than for HTTP cookies. A difference is that expiry dates are not supported in Local Storage and IndexedDB. Also, all measures in HTTP cookies related to HTTP (i.e. HTTP-only cookies) are impossible to implement at the level of HTML. However, secure HTTP cookies which can only be accessed using a encrypted HTTPS connection can be realized, because the protocol used is part of Same Origin Policy. Most other recommendations are implemented to a greater or lesser extend.

From all browsers we looked into Google Chrome and Mozilla Firefox have the best implementation. Only, Mozilla Firefox has the limitation that IndexedDB is not available in private browsing mode. A drawback both browsers have is that when private browsing mode is used, data is shared between multiple private tabs. Microsoft Internet Explorer's and Microsoft Edge's privacy settings do not always involve Local Storage and IndexedDB resources. Another problem that Internet Explorer and Edge have is that Same Origin Policy is not implemented fully according to the specification.

Because HTML5 storage techniques are treated the same as HTTP cookies, the storage is also deleted when HTTP cookies are deleted. This means that Local Storage and IndexedDB cannot be used as a supercookie mechanism to restore cookies when they are deleted.

By default privacy is slightly better defended in HTML5 than with HTTP cookies, because a party must be allowed to execute JavaScript from the same origin as where its resources are stored. This makes tracking harder for third party tracking services. However, by using frames the protection mechanisms can be circumvented. Embedded frames of tracking services can communicate with the original website by putting information in URLs or using the `postMessage` API for cross frame communication.

## Recommendations

When the implementation of HTML5 storage techniques in browsers is adapted it has the potential to have better privacy measures than cookies. To realize this all communication between frames from a different origin should be prevented. Even this would not rule out user tracking via HTML5 storage techniques completely, because a parent frame always has control over the source URL of an embedded iframe. A solution could be to also add a whitelisting feature for Local Storage and IndexedDB usage. By this it becomes visible which parties want to access stored information and it can lead to more awareness among users that there are tracking parties present at a website.

Only making more strict privacy measures for HTML5 storing techniques is not sufficient, because from Figure 3.1 we know that in practice HTML5 is always used in combination with a HTTP or HTTPS connection. This means that HTTP cookies are always an available alternative. So for privacy measures to work, only limiting some storage techniques is not sufficient. It is not likely that HTTP will be changed to have a more privacy friendly cookie mechanism, because the protocol is one of the foundations of websites nowadays.

## Benefits for Tracking Parties

The benefits for tracking parties to use Local Storage and IndexedDB over HTTP cookies are not that big. It only creates some opportunities for tracking on web applications that do not always contact the server when the user interacts with it. For other sites the expectation is that they remain using cookies instead of HTML5 storage techniques with slightly more limitations and more overhead.

This expectation is also confirmed by the empirical analysis discussed in Chapter 5. Out of 40 domains only 6 domains use Local Storage for tracking

purposes. IndexedDB tracking usage was not found by one of our tests using the browser extension. We only found one example in our orientating phase (Section 5.1).

## Reflection

My personal experience with dealing with implementation recommendations, especially the sections about privacy, is that they leave many important implementation details to software manufacturers. Therefore just reading specifications and recommendations does not give the full picture of the situation. Some additional empirical analysis of pieces of software that implement the specification of recommendation was therefore a good choice in my opinion.

About the empirical analysis of websites to analyse the usage of Local Storage and IndexedDB for tracking purposes, I think it was useful to only look to already known tracking parties. Something that we did not do, but could have been useful is that we did not store the results of frames that did not belong to any of the chosen tracking services. Due to this we were not able to give results about the percentage of all frames that belong to one of the tracking services. Another welcome additional check would have been to also look for tracking services that do not use frames. Now we were not able to tell something about some of the services because we did not find anything about it. Very likely is that those services were present, but did not use frames. When we also checked this, we would have had more information.

I can recommend analysing tracking usage using browser extensions. My experience with Google Chrome extensions is that it is not very complicated to make an extension. With some basic web development skills in HTML and JavaScript you can get already very far. In the extension's manifest you can easily define what script must be executed where.

# Bibliography

- [1] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 674–689. ACM, 2014.
- [2] Alexa. Top 500 sites on the web. <http://www.alexa.com/topsites>, 2016. Retrieved May 29, 2016.
- [3] Alexa. Top sites in Netherlands. <http://www.alexa.com/topsites/countries/NL>, 2016. Retrieved May 29, 2016.
- [4] A. Barth and U.C. Berkeley. HTTP State Management Mechanism. RFC 6265, Internet Engineering Task Force, April 2011.
- [5] Tomasz Bujlow, Valentín Carela-Español, Josep Solé-Pareta, and Pere Barlet-Ros. Web tracking: Mechanisms, implications, and defenses. *arXiv preprint arXiv:1507.07872*, 2015.
- [6] William Chan. HTTP/2 considerations and tradeoffs. <https://insouciant.org/tech/http-slash-2-considerations-and-tradeoffs/#TLSPrivacy>, 2014. Retrieved June 27, 2016.
- [7] Nik Cubrilovic. Persistent and unblockable cookies using HTTP headers. <https://www.nikcub.com/posts/persistent-and-unblockable-cookies-using-http-headers/>, 2011.
- [8] Peter Eckersley. How unique is your web browser? In *Privacy Enhancing Technologies*, pages 1–18. Springer, 2010.
- [9] Marjan Falahrastegar, Hamed Haddadi, Steve Uhlig, and Richard Mortier. The rise of panopticons: Examining region-specific third-party web tracking. In *Traffic Monitoring and Analysis*, pages 104–114. Springer, 2014.

- [10] Ian Hickson. Web SQL database. Working group note, W3C, November 2010. <https://www.w3.org/TR/webdatabase/>.
- [11] Ian Hickson. HTML5 web messaging. Recommendation, W3C, May 2015. <https://www.w3.org/TR/webmessaging/>.
- [12] Ian Hickson. Web storage (second edition). Recommendation, W3C, November 2015. <https://www.w3.org/TR/2015/PR-webstorage-20151126/>.
- [13] Ian Hickson, Robin Berjon, Steve Faulkner, Travis Leithead, Erika Doyle Navara, Edward O'Connor, and Silvia Pfeiffer. HTML5. Recommendation, W3C, October 2014. <https://www.w3.org/TR/2014/REC-html5-20141028/>.
- [14] Samy Kamkar. Evercookie Javascript API. <https://github.com/samyk/evercookie>.
- [15] Stefan Kimak and Jeremy Ellman. The role of HTML5 IndexedDB, the past, present and future. In *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 379–383. IEEE, 2015.
- [16] Balachander Krishnamurthy and Craig Wills. Privacy diffusion on the web: a longitudinal perspective. In *Proceedings of the 18th international conference on World wide web*, pages 541–550. ACM, 2009.
- [17] Nikunj Mehta, Jonas Sicking, Eliot Graff, Andrei Popescu, Jeremy Orlow, and Joshua Bell. Indexed database API. Recommendation, W3C, January 2015. <https://www.w3.org/TR/2015/REC-IndexedDB-20150108/>.
- [18] Hassan Metwalley, Stefano Traverso, and M Marco. Unsupervised detection of web trackers. In *IEEE GLOBECOM*, 2015.
- [19] Hassan Metwalley, Stefano Traverso, and Marco Mellia. Using passive measurements to demystify online trackers. *Computer*, 49(3):50–55, 2016.
- [20] Keaton Mowery and Hovav Shacham. Pixel perfect: Fingerprinting canvas in HTML5. In *Proceedings of W2SP 2012*. IEEE Computer Society, May 2012.
- [21] Google Chromium Projects. NPAPI deprecation: developer guide. <https://www.chromium.org/developers/npapi-deprecation>.

- [22] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. Detecting and defending against third-party tracking on the web. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 12–12. USENIX Association, 2012.
- [23] Ashkan Soltani, Shannon Canty, Quentin Mayo, Lauren Thomas, and Chris Jay Hoofnagle. Flash cookies and privacy. In *AAAI Spring Symposium: Intelligent Information Privacy Management*, volume 2010, pages 158–163, 2010.
- [24] Sid Stamm. Clearing Flash cookies using Firefox. <http://blog.sidstamm.com/2011/05/clearing-flash-cookies-using-firefox.html>, 2011.
- [25] The PageFair Team. The 2015 ad blocking report. <https://blog.pagefair.com/2015/ad-blocking-report/>.
- [26] Anne van Kesteren. Cross-origin resource sharing. Recommendation, W3C, January 2014. <https://www.w3.org/TR/cors/>.
- [27] Patrick Verleg. Cache cookies: searching for hidden browser storage, 2014. Bachelor Thesis. Radboud University Nijmegen.
- [28] W3C. Same Origin Policy. [https://www.w3.org/Security/wiki/Same\\_Origin\\_Policy](https://www.w3.org/Security/wiki/Same_Origin_Policy), 2010.

## Appendix A

# Example: Data Storing with IndexedDB

This JavaScript code creates an IndexedDB database and adds the JSON string `testData` (`{id: "42", test_string: "Hello world!"}`) to the database. The function `remove` deletes the database again.

```
window.indexedDB = window.indexedDB || window.mozIndexedDB ||
    window.webkitIndexedDB || window.msIndexedDB;

window.IDBTransaction = window.IDBTransaction || window.
    webkitIDBTransaction || window.msIDBTransaction;
window.IDBKeyRange = window.IDBKeyRange || window.
    webkitIDBKeyRange || window.msIDBKeyRange;

if (!window.indexedDB) {
    window.alert("Your browser doesn't support a stable version
        of IndexedDB.")
}

const testData = {id: "42", test_string: "Hello world!"};

var db;
var request = window.indexedDB.open("TestDatabase", 1);

request.onerror = function(event) {
    console.log("error: database could not be opened.");
};

request.onupgradeneeded = function(event) {
    var db = event.target.result;
    var objectStore = db.createObjectStore("testObject", {
        keyPath: "id"});
}

request.onsuccess = function(event) {
    db = event.target.result;
```

```

console.log("database has been opened succesfully.");

var request = db.transaction(["testObject"], "readwrite")
    .objectStore("testObject").add(testData);

request.onsuccess = function(event) {
    alert("Test object has been added succesfully to
        the database.");
};

request.onerror = function(event) {
    alert("Unable to add data\r\nObject does already
        exist in the database!");
}

db.close();
};

function remove() {
    var req = indexedDB.deleteDatabase("TestDatabase");
    req.onsuccess = function () {
        console.log("Deleted database successfully");
    };
    req.onerror = function () {
        console.log("Could not delete database");
    };
    req.onblocked = function () {
        console.log("Could not delete database due to
            the operation being blocked");
    };
}

```

## Appendix B

# Visited Websites for Empirical Analysis

All websites that we visited to collect the usage information of tracking companies using the developed chrome extension discussed in Section 5.2.3 are listed below:

- google.com
- youtube.com
- yahoo.com
- wikipedia.org
- amazon.com
- qq.com
- speedtest.net
- bing.com
- telegraaf.nl
- nu.nl
- tweakers.net
- ad.nl
- nos.nl
- rabobank.nl
- ing.nl
- abnamro.nl
- ebay.com
- marktplaats.nl
- ikea.com
- reddit.com
- imgur.com
- microsoft.com
- apple.com
- oneclickads.net
- stackoverflow.com
- alibaba.com
- cnn.com
- adobe.com
- nytimes.com
- bbc.com
- buienradar.nl
- dg.nl
- vi.nl
- walmart.com
- weather.com
- theguardian.com
- thetimes.co.uk
- abcnews.go.com
- ziggo.nl
- kpn.com
- dailymotion.com
- skyscanner.nl

## Appendix C

# Chrome Extension Source Code

In this research we used a Google Chrome browser extension to analyse webpages on LocalStorage and IndexedDB usage by tracking services (Chapter 5). The extension is named Tracking Trackers. The source code of the three most important scripts are attached here. The complete extension can also be found on GitHub: <https://github.com/ivard/TrackingTrackers>.

### C.1 check\_current\_frame.js

```
/*
  This content script is injected in every frame present on
  the current web page.
  If the source URL of the frame contains to one of the strings
  in trackers as substring,
  the frame is identified as belonging to a chosen tracking
  service. Then the usage
  statistics are communicated to the background script (see
  background.js).
*/
trackers = [
  'doubleclick ',
  'google-analytics ',
  'googlesyndication ',
  'googleadservices ',
  'scorecardresearch ',
  'imrworldwide ',
  'adnxs ',
  'criteo ',
  'rubiconproject ',
  'serving-sys ',
  'adform ',
  'googletagservices ',
  'googletagmanager ',
```

```

    'pubmatic',
    'ads.yahoo',
    'addthis',
    'turn.com',
    'bluekai',
    'mathtag',
    'adadvisor',
];

for(t in trackers) {
    if (document.domain.indexOf(trackers[t]) != -1) {
        var localStorageUsed = localStorage.key(0) != null;
        chrome.runtime.sendMessage({type: 'tracker', tracker:
            document.domain, localStorageUsed: localStorageUsed,
            indexedDBUsed: false}, function(resp){
            //alert(resp); //DEBUG LINE
        });

        indexedDB.webkitGetDatabaseNames().onsuccess = function(
            sender, args) {
            var indexedDBUsed = sender.target.result.length > 0;
            chrome.runtime.sendMessage({type: 'tracker', tracker
                : document.domain, localStorageUsed: false,
                indexedDBUsed: indexedDBUsed}, function(resp){
                //alert(resp); // DEBUG LINE
            });
        };
    }
}

```

## C.2 background.js

```

/*
This script is the main script belonging to the chrome extension
itself.
It receives the data from all content scripts injected into web
pages and
stores the data.

```

### IMPORTANT:

- The data will not be preserved if the browser is closed by the user!
- The extension only works correctly when using one browser tab at the same time. Otherwise it might happen that frames are matched with another active browser tab.
- Always wait before a website is fully loaded. Otherwise you might miss results. This browser extension does not enforce this!

```

current_url stores the url of the tab it is currently storing
data of.
The data is updated when the user navigates to another website
or moves to
another tab.
*/
var current_url = '';

function update_url() {
    chrome.tabs.query({
        active: true,
        lastFocusedWindow: true
    }, function (tabs) {
        current_url = tabs[0].url;
    });
}

chrome.tabs.onUpdated.addListener(function(tabId, changeInfo,
    tab) {
    update_url();
});

chrome.tabs.onCreated.addListener(function(tabId, changeInfo,
    tab) {
    update_url();
});

/*
saved_tracker is a dictionary that for every domain belonging to
a tracking services
stores which websites used it. For each of those websites is
stored whether Local Storage
and/or IndexedDB were used by the particular tracking domain.

Structure:
{
    tracker-a.com => [{url: website-a.com, localStorageUsed:
        true, indexedDBUsed: false}, ...]
    tracker-b.com => ...
}
*/
var saved_trackers = {};

chrome.runtime.onMessage.addListener(function(request, sender,
    sendResponse) {
    chrome.tabs.query({
        active: true,
        lastFocusedWindow: true
    }, function (tabs) {
        current_url = tabs[0].url;
        update_saved_trackers(request);
    });
});

```

```

function update_saved_trackers(request) {
    // First check type of request to make other type of
    // requests possible in the future.
    if (request.type == 'tracker') {
        if(saved_trackers[request.tracker] != undefined) {
            // Tracker domain has been found earlier
            sites = saved_trackers[request.tracker];
            var site_already_found = false;
            for(s in sites) {
                if(current_url == sites[s].url) {
                    // Specific website has been visited before,
                    // update data.
                    sites[s].localStorageUsed = sites[s].
                        localStorageUsed ? true : request.
                        localStorageUsed;
                    sites[s].indexedDBUsed = sites[s].
                        indexedDBUsed ? true : request.
                        indexedDBUsed;
                    site_already_found = true;
                }
            }
            if(!site_already_found) {
                // New website found that uses already known
                // tracking domain
                sites.push({url: current_url, localStorageUsed:
                    request.localStorageUsed, indexedDBUsed:
                    request.indexedDBUsed});
            }
            saved_trackers[request.tracker] = sites;
        }
        else {
            // New tracking domain found
            saved_trackers[request.tracker] = [{url: current_url
                , localStorageUsed: request.localStorageUsed,
                indexedDBUsed: request.indexedDBUsed}];
        }
    }
}

```

### C.3 popup.js

```

/*
    This script generates table with all found information
    stored by the
    background script.
*/

var saved_trackers = chrome.extension.getBackgroundPage().
    saved_trackers;
var table = document.getElementById("current_trackers");
for (key in saved_trackers) {
    var tracker_row = table.insertRow(-1);
    var tracker_col = tracker_row.insertCell(-1);

```

```

tracker_col.rowSpan = saved_trackers[key].length;
tracker_col.innerHTML = key;
var site_row = tracker_row;
for(s in saved_trackers[key]) {
    site_row.insertCell(-1).innerHTML = saved_trackers[key][
        s].url;
    site_row.insertCell(-1).innerHTML = saved_trackers[key][
        s].localStorageUsed ? "Yes" : "No";
    site_row.insertCell(-1).innerHTML = saved_trackers[key][
        s].indexedDBUsed ? "Yes" : "No";
    site_row = table.insertRow(-1);
}
table.deleteRow(-1);
}

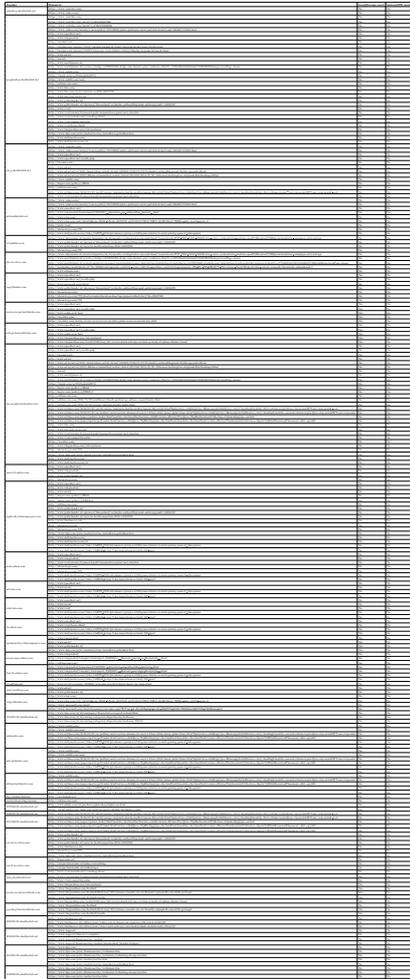
// Code to enable page to be opened in a separate browser tab.
document.addEventListener('DOMContentLoaded', function() {
    var link = document.getElementById('launch');
    link.addEventListener('click', function() {
        chrome.tabs.create({url: '/popup.html'});
    });
});

```

## Appendix D

# Testing Results of Empirical Analysis

Table is included small to fit on one page.

The table is extremely small and dense, consisting of many rows and columns. The text within the cells is illegible due to the small size. It appears to be a large data table with multiple columns and many rows of data.