UNIVERSITÄT PASSAU
Fakultät für Informatik und Mathematik

Dissertation

Preventing the Leakage of Privacy Sensitive User Data on the Web

Martin Koop (geb. Stopczynski)
Dissertation eingereicht an der Fakultät für Informatik und Mathematik der Universität Passau zur Erlangung des Grades eines Doktors der Ingenieurswissenschaften

Gutachter:	Prof. Dr. Stefan Katzenbeisser
Zweitgutachterin:	Prof. Dr. Delphine Reinhardt
Betreuer:	Prof. Dr. Stefan Katzenbeisser

Roßdorf, September 2020

Zusammenfassung

Das Aufzeichnen der Internetaktivität ist mit der Verknüpfung persönlicher Daten zu einer Schlüsselressource für viele kostenpflichtige und kostenfreie Dienste im Web geworden. Diese Dienste sind zum einen Webanwendungen, wie beispielsweise die von Google bereitgestellten Karten/Navigation oder Websuche, die täglich kostenlos verwendet werden. Zum anderen sind es alle Webseiten, die meist kostenlos Nachrichten oder allgemeine Informationen zu verschiedenen Themen bereitstellen. Durch das Aufrufen und die Nutzung dieser Webdienste werden alle Informationen, die im Webdienst verarbeitet werden, an den Dienstanbieter weitergeben. Dies umfasst nicht nur die im Benutzerkonto des Webdienstes gespeicherte Profildaten wie Name oder Adresse, sondern auch die Aktivität mit dem Webdienst wie das anklicken von Links oder die Verweildauer.

Darüber hinaus gibt es jedoch auch unzählige Drittparteien, welche zumeist im Hintergrund in die Webdienste eingebunden sind und das Benutzerverhalten der kompletten Webaktivität - Webseiten übergreifend - mitspeichern sowie auswerten. Der Einsatz verschiedener, in der Regel für den Benutzer verborgener Techniken, dient dazu das Online-Verhalten der Benutzer genau zu verfolgen und viele sensible Daten zu sammeln. Dieses Verhalten wird als Web-Tracking bezeichnet und wird hauptsächlich von Werbeunternehmen genutzt. Die gesammelten Daten sind oft personenbezogen und eine wertvolle Ressourcen der Unternehmen, um Beispielsweise passend zum Benutzerprofil personalisierte Werbung schalten zu können. Mit der Nutzung dieser personenbezogenen Daten entstehen aber auch weitreichendere Auswirkungen, welche sich unter anderem in Preisanpassungen für Benutzer mit speziellen Profilattributen, wie der Nutzung von teuren Endgeräten, widerspiegeln.

Ziel dieser Arbeit ist es die Privatsphäre der Nutzer im Internet zu steigern und die Nutzerverfolgung von Web-Tracking signifikant zu reduzieren. Dabei stellen sich vier Herausforderungen, die jeweils einen Forschungsschwerpunkt dieser Arbeit bilden: (1) Systematische Analyse und Einordnung eingesetzter Tracking-Techniken, (2) Untersuchung vorhandener Schutzmechanismen und deren Schwachstellen, (3) Konzeption einer Referenzarchitektur zum Schutz vor Web-Tracking und (4) Entwurf einer automatisierten Testumgebungen unter Realbedingungen, um die Reduzierung von Web-Tracking in den entwickelten Schutzmaßnahmen zu untersuchen. Jeder dieser Forschungsschwerpunkte stellt neue Beiträge bereit, um einheitlich das übergeordnete Ziel zu erreichen: *der Entwicklung von Schutzmaßnahmen gegen die Preisgabe sensibler Benutzerdaten im Internet*.

Der erste wissenschaftliche Beitrag dieser Dissertation ist eine umfassende Evaluation eingesetzter Web-Tracking Techniken und Methoden, sowie deren Gefahren, Risiken und Implikationen für die Privatsphäre der Internetnutzer. Die Evaluation beinhaltet zusätzlich die Untersuchung vorhandener Tracking-Schutzmechanismen und deren Schwachstellen. Die gewonnenen Erkenntnisse sind maßgeblich für die in dieser Arbeit neu entwickelten Ansätze und verbessern den bisherigen nicht hinreichend gewährleisteten Schutz vor Web-Tracking.

Der zweite wissenschaftliche Beitrag ist die Entwicklung einer robusten Klassifizierung von Web-Tracking, der Entwurf einer effizienten Architektur zur Langzeituntersuchung von Web-Tracking sowie einer interaktiven Visualisierung des Auftreten von Web-Tracking im Internet. Dabei basiert der neue Klassifizierungsansatz, um Tracking zu identifizieren, auf der Entropie Messung des Informationsgehalts von Cookies. Die Resultate der Web-Tracking Langzeitstudien sind unter anderem 1.209 identifizierte Tracking-Domains auf den meistbesuchten Webseiten in Deutschland. Hierbei wurden innerhalb der Top 25 Webseiten im Durchschnitt 45 Tracking-Elemente pro Webseite gefunden. Der Tracker mit dem höchsten Potenzial zum Erstellen eines Benutzerprofils war *doubleclick.com*, da er 90% der Webseiten überwacht. Die Auswertung des untersuchten Tracking-Netzwerks ergab weiterhin einen detaillierten Einblick in die Tracking-Technik mithilfe von Weiterleitungslinks. Dabei haben wir 1,2 Millionen HTTP-Traces von monatelangen Crawls der 50.000 international meistbesuchten Webseiten analysiert. Die Ergebnisse zeigen, dass 11,6% dieser Webseiten HTTP-Redirects, verborgen in Webseiten-Links, zum

Tracken verwenden. Dies wird eingesetzt, um den Webseitenverlauf des Benutzers nach dem Klick durch eine Kette von (Tracking-)Servern umzuleiten, welche in der Regel nicht sichtbar sind, bevor das beabsichtigte Link-Ziel geladen wird. In diesem Szenario erfasst der Tracker wertvolle Verbindungs-Metadaten zu Inhalt, Thema oder Benutzerinteressen der Website. Die Visualisierung des Tracking Ökosystem stellen wir in einem interaktiven Open-Source Web-Tool bereit.

Der dritte wissenschaftliche Beitrag dieser Dissertation ist die Konzeption von zwei neuartigen Schutzmechanismen gegen Web-Tracking und der Aufbau einer automatisierten Simulationsumgebung unter Realbedingungen, um die Effektivität der Umsetzungen zu verifizieren. Der Fokus liegt auf den beiden meist verwendeten Tracking-Verfahren: Cookies (hierbei wird eine eindeutigen ID auf dem Gerät des Benutzers gespeichert), sowie Browser-Fingerprinting. Letzteres beschreibt eine Methode zum Sammeln einer Vielzahl an Geräteeigenschaften, um den Benutzer eindeutig zu (re-)identifizieren, ohne eine eindeutige ID auf dem Gerät zu speichern.

Um die Effektivität der in dieser Arbeit entwickelten Schutzmechanismen vor Web-Tracking zu untersuchen, implementierten und evaluierten wir die Schutzkonzepte direkt im Chromium Browser. Das Ergebnis zeigt eine erfolgreiche Reduzierung von Web-Tracking um 44%. Zusätzlich verbessert das in dieser Arbeit entwickelte Konzept “Site Isolation” den Datenschutz des privaten Browsing-Modus, ermöglicht das Setzen eines manuellen Speicher-Zeitlimits von Cookies und schützt den Browser gegen verschiedene Bedrohungen wie CSRF (Cross-Site Request Forgery) oder CORS (Cross-Origin Ressource Sharing). Site Isolation speichert dabei den Status der lokalen Website in separaten Containern und kann dadurch diverse Tracking-Methoden wie Cookies, lokalStorage oder redirect tracking verhindern. Bei der Auswertung von 1,6 Millionen Webseiten haben wir gezeigt, dass der Tracker *doubleclick.com* das höchste Potenzial besitzt, den Nutzer zu verfolgen und auf 25% der 40.000 international meistbesuchten Webseiten vertreten ist.

Schließlich demonstrieren wir in unserem erweiterten Chromium-Browser einen robusten Browser-Fingerprinting-Schutz. Der Test unseres Prototyps mittels 70.000 Browsersitzungen zeigt, dass unser Browser den Nutzer vor sogenanntem Browser-Fingerprinting Tracking schützt. Im Vergleich zu fünf anderen Browser-Fingerprint-Tools erzielte unser Prototyp die besten Ergebnisse und ist der erste Schutzmechanismus gegen Flash sowie Canvas Fingerprinting.

Abstract

Recording Internet activity and linking it to personal information has become a key resource for many paid and free web services. These services include web applications such as Google Maps/Navigation or Search, which are used daily for free. On the other hand, there are websites that provide (mostly free) news or general information on various topics. By using these web services, all information is processed by the service provider. This includes not only the profile data stored in the user account of the web service such as name or address, but also the activities such as clicking links or the duration of stay. In addition, however, there are also countless third-parties, mostly hidden in the background, that record and analyze the users' behavior even across websites. This process is known as web tracking and is mainly used by advertising companies. The collected data often contains personal data which is a valuable resource for companies, for example, to provide personalized advertisements according to the user profile. However, the use of these personal data also results in far-reaching effects, which are among other things, price adjustments for users with special profile attributes, such as the use of expensive end devices.

The main goal of this work is to increase the privacy of users on the Internet and to significantly reduce web tracking. There are four challenges, each of which forms a research focus of this work: (1) systematic analysis and classification of used tracking techniques, (2) evaluation of existing protection mechanisms and their weaknesses, (3) designing a reference architecture to prevent web tracking; and (4) designing an automated test environment under real-world conditions to evaluate the reduction of web tracking in the developed protection mechanisms. Each of these research topics provides new contributions to achieve the overall objective: preventing the leakage of privacy sensitive user data on the web.

The first scientific contribution of this dissertation is a comprehensive evaluation of web tracking techniques used on the web, as well as their risks, threats and implications for the users' privacy. The evaluation also includes the analysis of existing tracking protection mechanisms and their weaknesses. The insights gained are decisive for the new developed approaches in this work and improve the previously insufficient protection against web tracking.

The second scientific contribution is the development of a robust classification of web tracking, the design of an efficient architecture for a long-term study of web tracking and the publication of an interactive visualization of the occurrence of web tracking on the Internet. The new classification approach to identify tracking is based on the entropy measurement of the information content of cookies. The results of the web tracking long-term studies include 1,209 identified tracking domains on the most visited websites in Germany. Here, an average of 45 tracking elements per website was found within the top 25 websites. The tracker with the highest potential for creating a user profile was doubleclick.com, as it monitors 90% of the scanned web pages. The evaluation of the tracking network provided a detailed insight into the tracking technique using redirect links. In this process, we analyzed 1.2 million HTTP traces of months long crawls on the 50,000 internationally most visited websites. The results show that 11.6% of these websites use HTTP redirects hidden in web links capable for tracking. This method is used to redirected the user through a chain of potential tracking servers not visible to the user. In this scenario, the tracker collects valuable connection meta data about the content, topic or users' interests on the website. The visualization of the tracking ecosystem is provided in an interactive open-source web tool.

The third scientific contribution of this dissertation are the design of two novel protection mechanisms against web tracking and the development of an automated simulation environment under real conditions to verify the effectiveness of the implementations. Here, we focus is on the two most commonly used tracking methods: cookies (where a unique ID is stored on the user's device), as well as browser

fingerprinting. The latter describes a method for collecting a variety of device properties to uniquely (re-)identify the user without storing a unique ID on the device.

In order to evaluate the effectiveness of the web tracking protection mechanisms developed in this work, we implemented and evaluated the protection concepts directly in the Chromium browser. The result shows a successful reduction of web tracking by 44%. In addition, the concept of "Site Isolation" developed in this work improves the privacy of the private browsing mode, enables setting a manual cookie timeout limit and protecting the browser against various threats such as Cross-Site Request Forgery (CSRF) or Cross-Origin Resource Sharing (CORS). Site Isolation stores the status of the local website in separate containers and can therefore prevent various tracking methods such as cookies, lokalStorage or redirect tracking. In an additional evaluation of 1.6 million websites we show that the tracker doubleclick.com has still the highest potential to track the user on 25% of the 40,000 internationally most visited websites.

Finally, we demonstrate a robust browser fingerprinting protection in our enhanced Chromium browser. Testing our prototype on 70,000 browsing session, we show that our browser protects against so called browser fingerprinting tracking. Compared to five other browser fingerprinting tools, our prototype achieved the best results and is the first protection mechanism against Flash as well as Canvas fingerprinting.

Acknowledgments

During my long journey leading to this dissertation, there are various people I would like to thank. First of all, I would like to thank my advisor Prof. Dr. Stefan Katzenbeisser. Being my supervisor, he not only provided me continuous support by reviewing my work, but especially motivated me in my research with new ideas and his feedback leading to great publications. I thank him for his patience and the possibility he gave me to finish this work. I very well remember his inspiring goal he told me once "*to support all students finishing their studies*". I also thank my co-referee Prof. Dr. Delphine Reinhardt, giving me the confidence to finish the thesis.

Further, I thank all me friends in believing in me to finish my dissertation. I thank in particular my friends Franz Aschoff, Matthias Finke, Bastian Koop, Johannes Küber, Johannes Lerch, Neal Livingston, Kamill Panitzek and Lukas Stopczynski helping me to proofread the thesis. I especially thank my colleague and friend Marco Ghiglieri, giving me the entire time support in technical problems as well as in discussing research questions. In addition, I thank Erik Tews for his help in supporting my publications with ideas as well as technical know how. Moreover, I thank all my students making contributions to my research projects. Thank you Peter Baumann, Benedikt Bäumle, Thorben Bürgel, Johannes Häußler, and Michael Zugelder for the great work in your theses.

I would like to express my gratitude to my parents, as they gave me all the support I needed to start and finish my studies. All is based on the life goal of my family and what my grandfather used to say: "*It is better to carry books, than a sack of coal*", which helped me reaching this point.

Special thanks to Lara, giving me the energy and inspiration in the last steps finishing this thesis. You motivated me to accomplish my goal - I love you!

Contents

List of Figures

List of Tables

Listings

1 Introduction into Web Tracking

In the today's digital life we use a wide spectrum of online services, applications and tools. The age of the *Internet of Things* comprises a plethora of physical devises that collect and exchange data over the web. These devises such as computers, laptops, smart phones, smart TVs or smart watches are capable of browsing the web, using online services, communicate with each other as well as collect personal data and activities. Due to this interconnectivity of devises and the usage of web services as well as the exchange of data, users leave a huge digital footprint [1, 2, 3].

Users freely give away a lot of personal data just while browsing the web. This consists not only of the data which websites the users visit, what content and topics the users are interested in. Moreover, the data which physical devices and services the users are using or the content they are sharing in *Online Social Networks (OSN)* like Facebook[1], leaks a lot of privacy sensitive user data to the content provider as well as the embedded third-party services on the websites [4].

Connecting this data with other information from social media (Facebook *Likes*) [5, 1, 6], shopping behavior [7], mobile location information, physical devices [8, 9], search queries [10] and profile information in web services, more privacy sensitive information can be revealed [11, 12, 13, 14]. This includes personal problems or desires of users, real names, addresses, political or religious views, as well as the financial and health status [4, 15].

Depending on the users geographic location (computed using the IP address), the type of purchases, the devices the user is using and the websites visited, companies may then be able to create a personal credit score [16] or estimate the risk for health insurance to a disadvantage of the user [17, 18]. Moreover, the loss of anonymity can lead to price discrimination [19, 20], deductibility of private information [21] or identify theft/fraud.

As we will present in Chapter 3 and 5, our research [22, 23] depicts that user tracking is even more widespread in today's web then showed by other researchers [24, 25]. We show that third-party trackers exist on most commercial websites and are able to follow users on 90% of the top 1,634 mostly visited websites in Germany [22] as well as 25% of the global top 40,000 ranked websites [23]. Our research concludes that trackers are able to easily re-identify a user across the web [26, 24]. The tracking potential of the top trackers such as Google and Facebook is:

- 50% of the top 600 global pages [27],
- 25% of the top 10,000 global pages [28], and
- 10% of the top 1 million global pages [29].

1.1 Web Tracking Methods

In order to give an insight into web tracking, we will first briefly introduce the web tracking methods and techniques, followed by possible protection mechanisms and tools. A detailed discussion will be presented in Chapter 2.

It is common practice to use different tracking techniques by embedding third-party elements [13, 28]. This covers not only the web advertisement infrastructure like banners, but also so-called web beacons[2] or social media buttons, to gather data on the users' online behavior as well as privacy sensitive data [11, 12, 13, 30].

The goal is to establish detailed user profiles, which are then used for more effective targeted advertisements [31, 32, 33, 34], and even if anonymized, they can be reconstructed [35] or easily de-anonymized

[1] https://www.facebook.com

[2] Web beacon or web bug, are typically 1x1 pixel images embedded on a website and not visible to the user

and sold [14, 36, 37, 6]. Moreover, tracking is often invisible or integrated into other features and services, such as advertising, or the Facebook *Like* button. In effect, most tracking is performed in the background without the user's notice and there is usually no option to opt-out [12, 38]. Even with the EU's General Data Protection Regulation (GDPR), which should make the users aware about the general usage of cookies (introduced in the year 2018), tracking did not drop significantly as research shows [39, 40, 41].

In contrast to other kinds of user behavior tracking, for example via surveillance cameras or by evaluating the credit card history, web tracking is cheap, mostly automated and not limited to purchases done by credit card. It is also a lot faster and can be more reliable, especially when data can be linked to important and persistent online services [42], such as social media like Facebook or e-mail accounts, to create detailed user profiles [43].

The most common way for any company to track users on the web is by storing a unique identifier on the client and retrieve it on every page where the tracking code is embedded [44]. Unique identifiers can be stored for example in cookies, HTML5 localStorage, browser cache, and common third-party plug-ins like Flash or Java [13]. A basic method to save this identifier is by using a HTTP cookie. Any HTTP request suffices for storing a cookie on the users machine. The so called tracking pixel, a transparent 1:1 pixel sized image, is the traditional, bandwidth-conserving, and high-performance method used in the wild to set a cookie. Once set, the visitor's browser will request the tracking cookie each time a page is loaded and the tracking company can re-identify a user on every page that includes this tracking cookie.

Because users started to block or delete cookies, tracking companies started to use JavaScript to identify users by determining an almost unique browser fingerprint of the users device [30, 45, 46, 8, 47]. This is done by monitoring the mouse position and clicks, interactions with forms [48], retrieving the list of installed plug-ins or other hardware settings [49, 30]. Plug-ins such as Flash or Java even provide additional unique data, for example the list of installed fonts, CPU type or clock skew [50]. This technique is relies on uniquely identifying a browser or device through combined system information that are retrievable by websites using JavaScript, Java or Flash scripts [51], even without storing a custom identifier. The differences originate from the various hardware, network, or software configurations that a user's computer may have [45, 49, 52, 53, 54, 55]. This information, also called fingerprinting features, are again mainly used by companies to track user behavior on the web [30] and to establish detailed user profiles [24, 35].

As already presented by Mayer [56] and Eckersley [46] in 2009/2010, it was possible to uniquely identify 94% of 470,161 users by fingerprinting techniques. Further studies [57, 50] revealed the prevalence of browser fingerprinting in real world applications and showed that more sophisticated fingerprinting methods exist. Fingerprinters commonly use detailed information, such as system fonts, plug-in version numbers, or even differences in graphic rendering engines (using the HTML5 canvas element) as fingerprinting features [49].

1.2 Web Tracking Defenses

Protecting the privacy of web users against tracking by blocking third-party content has become a cat-and-mouse game. Continuously evolving tracking methods used by companies make it difficult to block all third-party content. In the last years many browser add-ons such as Privacy Badger[3], Adblock[4], Ghostery[5] or the Tor browser[6] were introduced to mitigate web tracking [25]. The capabilities of those anti-tracking tools differ, but the main functionality is often similar: block third-party connections on websites in order to stop displaying advertisements and prevent them from storing tracking elements, such as cookies, on the user's machine [38]. This includes third-party JavaScripts or other

[3] https://www.eff.org/de/node/73969
[4] https://getadblock.com/
[5] https://www.ghostery.com/
[6] https://www.torproject.org/

persistent identification elements stored on the user's machine. By blocking these third-party elements no connection to the supplying tracking server is established.

However, blocking third-party content can cause usability or functionality loss on websites if third-party images, Flash elements or JavaScripts cannot be loaded [38, 58]. Moreover, those anti-tracking tools are not sufficient to block all cookie based tracking techniques and are detectable by the trackers [24, 46, 59, 60, 49, 61, 62, 63]. Similar problems have protection tools blocking advanced tracking techniques such as browser fingerprinting [50, 51, 25, 10] or tracking redirect links, as we show in this work. Similarly, with the loss of revenue trough ad-blockers not displaying advertising [64], many websites implement anti ad-blocker scripts, forcing the user to disable their ad-blocker [65, 66, 67] or use other techniques such as redirect tracking. Further, elements not categorized as trackers will not be blocked and can still be used for tracking. Also, if legitimate elements are falsely classified (false positives) as trackers and blocked, web site functionalities may break. The most serious privacy problem exist in the commercial purpose of such anti-tracking tools, i.e., a third-party can avoid that certain tracking elements will be blocked (by making a contract) [61, 68], or by the utilization of the gathered user data [69] to allow building a user web profile.

1.3 Publications and Contributions

This work is based on the following publications:

The in-depth evaluation of redirect tracking and link usage [70] provides the first large scale analysis in this field, scanning 50,000 Alexa top ranked websites for several months. We extended a scanning framework with automated user interaction to generate a much more realistic browsing behavior. Analyzing 1.2 million redirect traces, we identified redirect tracking on 11.6% of those websites. Further, we developed an interactive web tool to visualize the tracking ecosystem and its network. Moreover, we present a redirect tracking network clustering and provide protection strategies against redirect tracking.

Our survey on web tracking [22] presents an extensive list on web tracking techniques and methods as well as their risks, threats and implications. We also discuss available protection mechanisms, tools and their weaknesses in protecting the user against web tracking. Moreover, we designed and imple-mented an automated system to detect, analyze and protect against web tracking. We evaluated the system in a 16 months long term web tracking study, scanning the top 1,634 mostly visited websites in Germany. In this study we detected 1209 trackers and exported these tracking domains as blocking list for the Internet Explorer, known as *Tracking Protection List*. In addition, we establishing the website www.trackyourtracker.de to visualize our results to the public.

A key contributions of this work are the design, implementation and evaluation of two systems providing protection mechanisms against web tracking. First, "Site Isolation" [23], which is a system to isolate the locally stored website state into separate containers. Site Isolation protects the user successfully against the tracking methods using: cookies, HTML5 localStorage, Index DB, browsing cache, Flash LSO and redirect tracking. In addition, we implemented an improved privacy feature for private browsing and added a cookie timeout feature in the browser. Moreover, Site Isolation also adds security features to protect against CORS, CSRF and click-jacking, cache-timing attacks and rendering engine hijacking automatically into the browser. In the evaluation on 1.6 million websites we then show that Site Isolation reduces web tracking by 44%.

Second, "Disguised Chromium Browser" [71], which was the first robust protection mechanism against Flash and canvas fingerprinting. Within the implementation on to different browser fingerprinting protection mechanisms, we developed a novel approach preventing canvas fingerprinting. In the evaluation, compared against five other anti-fingerprinting tools, we show in 70,000 browser sessions that our system provides a 99% protection against browser fingerprinting.

Additional Publications

Not covered in this thesis are the following publications, which were published during the period of the PhD:

- SecLab: An Innovative Approach to Learn and Understand Current Security and Privacy Issues
  Marco Ghiglieri, Martin Stopczynski
  In: SIGITE '16 Proceedings of the 17th Annual Conference on Information Technology Education. Pages 67-72, September 2016. ISBN 978-1-4503-4452-4

- Personal DLP for Facebook
  Marco Ghiglieri, Martin Stopczynski, Michael Waidner
  In: IEEE: Pervasive Computing and Communications Workshops (PERCOM Workshops), p. 629 - 634, March 2014

- Smart Home Dashboard – Das intelligente Energiemanagement
  Martin Stopczynski, Marco Ghiglieri
  In: VDE KONGRESS 2012 Smart Grid, VDE Verlag GmbH, November 2012. ISBN 978-3-8007-3446-7. www.vde-verlag.de/proceedings-de/453446061.html

- C4PS: colors for privacy settings
  Thomas Paul, Martin Stopczynski, Daniel Puscher, Melanie Volkamer, Thorsten Strufe
  In: ACM: Proceedings of the 21st international conference companion on World Wide Web, vol. WWW '12 Companion, p. 585–586, May 2012. ISBN 978-1-4503-1230-1. doi.acm.org/10.1145/2187980.2188139

- C4PS – Helping Facebookers Manage Their Privacy Settings
  Thomas Paul, Martin Stopczynski, Daniel Puscher, Melanie Volkamer, Thorsten Strufe
  In: Springer Berlin Heidelberg: Lecture Notes in Computer Science, Social Informatics, no. 7710, p. 188-201, 2012. ISBN 978-3-642-35385-7

1.4 Outline

In this thesis, we provide a detailed description of the tracking ecosystem and present a comprehensive web tracking background in Chapter 2. Here, we will first summarize the possible risks, threats and implications of web tracking. Subsequently, we introduce methods and techniques to track the users' online behavior. Moreover, we present the protection mechanisms and tools against web tracking as well as describe their weaknesses.

To get an insight in the tracking practice, we developed a general detection and prevent system for web tracking covering the most visited German websites. In Chapter 3 we give an overview of the system architecture and present the findings on our 16 months long term evaluation of web tracking on most visited German websites.

Throughout our research we identified the usage of redirect links as a tracking method. In Chapter 4 we present the first in-depth study on tracking potential of redirect links, in which trackers use different HTTP redirect techniques for tracking. Mostly hidden in website links, trackers detour users "intended" connection (link destination) through a chain of (third-party tracking) servers, before loading the intended (legitimate) link destination. This enables trackers on each redirected page to circumvent third-party blocking tools by storing first-party tracking elements on the users machine without notice. Compared to third-party cookies, trackers use the fact that first-party cookies are almost always enabled and required by websites to handle sessions. In addition, by using redirect links the trackers can collect valuable meta data such as the site of origin, the desired destination target, the topic/content the user is viewing and other information.

Focusing on user privacy, we give a comprehensive insight in the behavior, patterns and the used redirect link tracking techniques in the wild. Our dataset is based on full HTTP traces browsing the top 50,000 websites from the *Alexa.com* ranking[7] over a period of several months. With our advanced simulated user interaction, we provide a much more realistic user browsing behavior dataset as well as a broad data collection of tracker interaction as other studies. In this evaluation we point out the most common redirect trackers currently used, classify them and suggest countermeasures that will help to protect users from redirect trackers beyond the capabilities of the current privacy protection add-ons.

Moreover, we use tracker classification, taking third-party appearance in the HTML source code into account, and analyze the information content in cookies leading to realistic tracking classification. We also compare our results to other tracking classifications such as Easylist and OpenWPM [29] and visualize the performance as well as various tracker behaviors to the user in our interactive web graphs[8]. In addition, we analyze the behavior of link usage and tracking in the different browser cookie settings compared to using an ad-blocking tool.

Because of the privacy concerns as well as tracking risks described in this work, most users do not approve the collection of privacy sensitive data being collected [72, 73, 74]. Further, the weaknesses in the available tracking protection mechanisms we discovered, a fundamental goal of this work was to provide a robust tracking prevention system. Therefore, we implemented and successfully evaluated the first automatic isolation of the locally stored website state into separate containers in Chapter 5. Our approach eliminates the ability of trackers to re-identify users across different sites, by isolating HTTP cookies, HTML5 Web Storage, Indexed DB, and the browsing cache. The so-called *Site Isolation* was implemented for the Chromium browser and in addition secures the browser against CORS, CSRF, and click-jacking attacks, while limiting the impact of cache timing, and rendering engine hijacking. We evaluated the effectiveness of *Site Isolation* [23] by visiting 1.6 million pages on over 94,000 distinct domains and compared the data saved against usual browsing. We show that top trackers collect enough information to identify billions of users reliably. In contrast, with *Site Isolation* in place the number of tracked pages can be reduced by 44%.

Protecting the user against fingerprinting, existing tools rely on simply randomizing system parameters or blocking and deactivating specific features. Traditional methods lack in usability when browsing websites, are detectable by the trackers [30, 59, 75, 46, 76, 77], or have flaws making the user still identifiable. Motivated by those challenges, we implemented and evaluated robust methods protecting the user against browser fingerprinting in a *Disguised Chromium Browser* (DCB), presented in Chapter 6. Comparing other anti-fingerprinting tools, our solutions overcome existing flaws through enhanced protection mechanisms. Instead of disabling or randomizing system and browser parameters like other tools, we use a large set of real world parameters that are fixed for the entire browsing session and change automatically for next session. This prevents detectability through unrealistic or constantly changing system parameters, if a fingerprinter requests a parameter multiple times. In addition, we are the first to effectively protect against Flash as well as canvas fingerprinting without deactivating these features.

Through an intense study of canvas fingerprinting, we developed a novel and deterministic approach to prevent canvas fingerprinting based on randomly modifying the canvas element in every browsing session. It is not based on adding patterns or noise to the original canvas like in other tools. Because no original (unique) canvas element is returned, the fingerprinter is not able to detect the modification, therefore is not able to remove/delete any changes and is not able to identify any user uniquely. We show the effectiveness of the algorithm and demonstrate perfect diversity of every generated canvas element. In contrast to other anti-fingerprinting tools, we also implemented a new and undetectable protection mechanism against the retrieval of system fonts via Flash.

Our evaluation against real world fingerprinting tools demonstrates that fingerprinters cannot notice the presence of our counter-fingerprinting techniques. Since changes are applied inside the browser

[7] `http://s3.amazonaws.com/alexa-static/top-1m.csv.zip`
[8] https://tracking.dbvis.de/

itself and never reveal one's own system settings, but output other real world parameters, it is hard for fingerprinters to circumvent or revert those changes, like suggested in [78].

In Chapter 7 we summaries our findings and conclude with future work as well as recommendation how prevent the leakage of privacy sensitive data on the web.

2 Web Tracking Background

This chapter presents a detailed evaluation of web tracking as well as its ecosystem and is based on our publication Web Tracking Report [22]. We will review the fundamentals of web and cross-domain tracking as well as give an overview on the market and stakeholders. In this chapter, we will point out the risks, threats and implications of web tracking for the user. Summarizing the possible methods and techniques to track the users' online behavior in Section 2.6, we also gather the protection mechanisms as well as tools against web tracking. Moreover, we will show the weaknesses and drawbacks of the used protection mechanisms in Section 2.7.

2.1 Beginning of Web Tracking

While browsing the web, it is common practice to monitor the users' actions. This is legitimate by the technical perspective of the websites' owner. Through the request in displaying the website, all content, even embedded external (third-party) elements, are downloaded from a server and transmitted to the user. This includes storing cookies, for example to save the users online shopping items in a private cart through the browsing session. Since the server needs to process the users request, it will track the users actions. The same tracking can also by done by other (third) parties when the elements are embedded on the visited website. Embedding third-party elements is done intentionally by the website owner and can have multiple purposes. Third-party elements are used to implement specific website features like external guest-books, chats, online social media buttons, libraries for visual effects or embedding content that does not need to be stored on the own server, like movies, pictures as well as advertisements.

During the history of the web, the industry noticed the rising business market in placing advertisement on websites. In using the free Internet content users did not pay attention to ads in the beginning. Website owners on the other hand took their chance to reduce operational costs through additional financing opportunities with advertising. So the general opinion emerged, that the placement of advertisements seem not to do any or no harm, since nobody needs to look or click on it. However, by embedding third-party elements such as advertising or other content, the advertiser is able to view, collect and evaluate various information on the user when visiting the website and automatically downloading the content [79]. In consequence, this can create privacy threats to the users. Using different mechanisms, discussed in Section 2.6, third-parties are able to track the users interests in real time across many websites they are embedded on. We will show in Section 2.2, that the amount of trackers can (re-)identify users on about 25-90% of websites [28, 27]. Third-party tracking is mostly done in the background and without the users notice. The user has no idea what kind of data is gathered and has usually no option to opt-out. The loss of control of gathering privacy sensible information through third-parties is known as *web tracking* and is the topic of this work.

2.2 Cross-Domain Tracking

Tracking companies, in the following trackers, have the goal to establish a detailed and accurate user behavior profile. This is accomplished through the amount of information gathered by following the users on as many websites as possible. This tracking behavior is called *cross-domain tracking*, or sometimes *Re-targeting*. Here, the tracker is able to connect different user online behavior data such as the topics and content of websites the user is visiting, in order to create a specific user profile. With this profile it is possible to identify the users potential interests, political views, financial status and other information to serve the best fitting advertising [80, 81, 82, 33, 83, 34].

To track users across domains or devices, the tracker needs to re-identify the user. The primitive method is by using tracking cookies embedded in advertising or hidden as part of the received website

content, which we will show in Chapter 5. Additional user data can also be bought from other trackers or shared through ad networks [84, 3]. Tracking techniques and methods will be described in detail in Section 2.6.

An international study [28] on the top 10,000 websites identified that 46% of the websites contain at least one tracker, and that Google's doubleclick tracker is embedded in 25% of those 10,000 websites.

User tracking is not only limited to the Internet. Similar techniques are used on Smart TVs and Smartphone Apps, but are not part of this work.

2.3 Market and Stakeholders

Web tracking with online advertising is a profitable market. The revenues in online ads, which creates and uses user tracking information, are immense and reached in 2011 $80 billion US-Dollar in global spending [85]. In 2018 the digital and mobile advertising sales grew globally to $232 billion, expecting constant rise to $427 billion by the year 2022 [86]. Digital advertising sales outranked TV ad sales in 2017, reaching a 41% market share (compared to 35% for linear television) [87]. By 2020, the share of digital advertising will reach nearly 50% in total.

Search advertising remains the largest market in online advertising with 40% (around 56 billion revenue) [88] and its market leader is Google with over 50% [89]. But ad space in social media such as Facebook and Twitter changed this by the end of 2016. In addition, *ZenithOptimedia* expects that the main driver in global online ad spending will be the mobile market, gaining up to 70% market size in Internet advertising [90].

Ad placement and user tracking is a lucrative market. Many websites make money by placing ads or tracking elements [91, 63]. Here, users not only get specific content, but also personalized advertising [92]. Products are more focused on users interests, so the probability rises in making the deal [33]. As an example, when searching for a specific item like sport shoes, it is most likely to see sport shoe advertising on the next visited websites. This is what companies are willing to invest and pay extra [93, 94, 38]. According to [95] personalized ads are a factor of 670% more successful. This leads to a rise of personal utilized ads of 15% [96]. The advantage comparing to TV or print advertising is the possibility of evaluating the audience and success.

The advertising business created its own ecosystem. With advertising companies and advertising networks acting like agents, re-marketing ad-space on many websites. Being able to distribute real time ads on different platforms from a pool of different ads [97, 15]. Top ranked global ad networks are:

- Google Adsense / Adwords / Doubleclick
- Facebook Ads
- Baidu
- Microsoft Ads
- Yahoo / Verizon (AOL)
- Twitter

Out of 4 billion Internet users, Facebook for example has a reach of 50% with 2 billion monthly active users and could increase their advertisers spending to $40 billion (90% of those ads are mobile) [98]. This makes Facebook together with the leader Google covering about 60% of the U.S. ad market revenue [99].

The US company *Rapleaf* was known to tie personal data to email addresses, and selling the information of a specific attribute such as gender, interests or creditability. Claiming to have 80% of US email addresses, companies could buy information for an attribute of a users, which is connected to the users

email address for one cent. Most companies are interested in user information to increase the awareness of users with similar profiles in order to gain new customers [96]. To satisfy this rising demand [100], new companies develop more sophisticated methods and techniques in tracking users [72], which will be presented in Section 2.6.

2.4 Ad-Ecosystem

Displaying advertisements on websites started as basic banners, similar to ads in newspapers or magazines, fitting a specific topic and a user group that might be interested. Later, the ads are matched with search results targeting specific topics the user was looking for. Next, contextual advertising opened the field of automatically targeting the topic/context of an website, using search algorithms based on word meanings, built upon an underlying lexicon called WordNet. Then advertising platforms evolved to adapt a richer media environment, such as video, audio and mobile networks with geographical information. Around 2005, new platforms focused on real-time ad displays within pop-ups. Selling the appearance of ads were they where viewed, depending on the users actual interests. Unlike traditional ad networks, these ad exchanges need to aggregate multiple ad networks in order to deliver the best matching ad.

This challenge of finding the best match between a given user in a given context and available ads, is referred to as computational advertising. The best match, however, is not limited to the ”relevance” from the traditional informational retrieval research sense, but also includes the best revenue from the economic perspective. For instance, in the context of sponsored search, the challenge is to find and display the best ads from advertisers which suit user's interest (relevance) as well as generating as much revenue as possible [101]. Solving this challenge depends on computing power, algorithm design and available data. Companies like Google with a huge user base, data pool as well as computational power are therefore in a leading position.

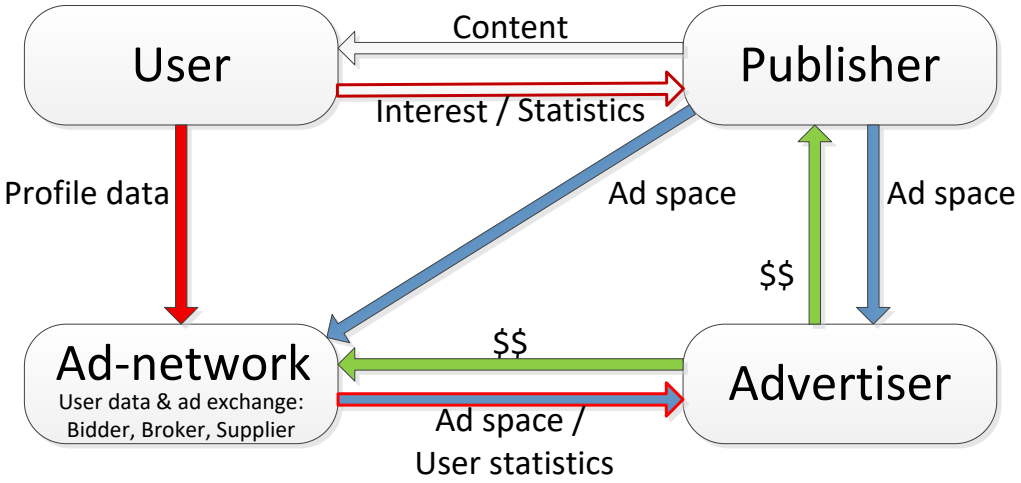

Figure 2.1: Ad-ecosystem

The general view of the ad-ecosystem is displayed in Figure 2.1, with main participants: User, publisher, advertiser and ad network. In this ecosystem the advertiser wants his ads to be displayed and is spending a budget to buy ad space from ad-networks or directly from publishers. A publisher provides content for the user and ad space to sell in order to gain revenue. The ad networks are bringing both parties together, serving as matcher for ads and available space on various publishers to reach the best performance. Using the publishers' service, users spend time on their website, looking at ads as well as interacting with them. The critical part is matching the ads by their keywords that are most relevant to

the user (based on the data gathered), or, if no user data is available, fitting the ad into the websites content or search query.

In the last years more platforms, services and companies are engaging in the business and provide new tools to optimize the revenue. Ad networks are becoming a complex structure of various platforms which contain:

- **Demand Side Platforms (DSP):** Ad agencies serve advertisers by bidding for their campaigns in multiple ad networks automatically.
- **Supply Side Platforms (SSP)**: Ad networks serve publishers by registering their ad space in their ad network.
- **Ad Exchanges Services (AES)**: AES combine multiple ad networks together. When publishers request ads with a given context to serve users, the AES contacts candidate Ad Networks (ADN) in real-time for a wider selection of relevant ads.
- **Data Management Platforms (DMP)**: Brokers serve DSP, SSP and AES by providing user profile data in real-time for better matching (Examples: DoubleClick, Invite Media, AppNexus, MediaMath).

The emergence of DSP, SSP, AES and DMP is a result of the fact that there are hundreds of ad networks available on the Internet, which can be a barrier for advertisers as well as publishers when getting into the online advertising business. Advertisers have to create and maintain campaigns frequently for better coverage, and analyze data across many platforms for a better impact. Publishers have to register and compare several ad networks carefully to achieve optimal revenue. The AES came as an aggregated marketplace of multiple ad networks to help alleviate such problems. Advertisers can create their campaigns and set desired targeting only once and analyse the performance data stream in a single place, and publishers can register with AES and collect the optimal profit without any manual interference [101].

But the borderline between these platforms is becoming less tangible. The DMP collects user data and sells it anonymously to DSP, SSP, AES as well as advertisers directly in real-time bidding (RTB) for better matching between ads and users. This technology is usually referred to as behavior targeting. Intuitively, if a user's browsing history shows interest in advertisers' products or services, then advertisers have a higher chance of securing a transaction by displaying their ads, which results in higher bidding for the impression. Initially the DMP was a component of other platforms, but now more individual DMPs are operating alongside analyzing and tracking services [101].

2.5 Risks, Threats and Implications

By the time new web features and technologies were developed, tracking became more sophisticated to uniquely identify users, their online behavior and the users' sensitive private data. Users tracking data is valuable and profitable to create more effective targeted advertisements [31, 32, 33, 34], in particular when the data can be linked with real world identities including the name, address and other personal data.

There exist different methods companies use to obtain detailed user data, resulting in various threats and implications for users that will be presented in this section.

2.5.1 Insecure Data Handling

Threats for users already arise in the insecure handling of their sensitive data by the service provider. This is the case when private data is stored insecurely, or is sent over insecure channels, not encrypted,

or encrypted with a weak algorithm or flawed implementation. Here, the data is at risk. For example, Motorola sent the login data to popular services such as Youtube and Facebook used by the user on the mobile phone to its own servers [102]. In this scenario the customers' email addresses were put into URL parameters of an external http resource. This data could be recorded and evaluated by attackers in the local network, the Internet Service Provider and various intelligence services. If the data is transmitted securely, e.g., using state of the art protocols without known weaknesses, tracking companies can still be attacked directly. These large collections of user data tend to attract external attackers, who routinely obtain mostly unsecured data sets containing data of millions [103].

2.5.2 Web Services and Social Media

In the recent years many new web services and social media arise. Services such as *Facebook.com*, *Twitter.com*, *Flickr.com*, *Pinterest.com*, *Digg.com* and others provide a platform for users to share information, pictures or thoughts with others. In addition to publishing users uploaded content, these web services developed interfaces (social media buttons) that can be included on other websites to share (publish) the websites content in the users own social media account. Using the social button, users can share website articles, videos or images they saw online with their friends and followers with one click. By doing so, the content and meta data can be used by the service provider (e.g., Facebook) to derive the users interests or desires and establish detailed user profiles [104]. This data is then sold indirectly, as a package of advertising targeted to a specified group of users, e.g., Facebook [105], or sold in anonymized form, e.g., by Acxiom[1] and Nielsen[2].

The website embedding those social media buttons on the other hand can get the visitor profile data in return. Using the personal user data the website can provide personalized ads, which will be rewarded by the advertising companies with money [92, 106].

Connecting the gathered data from search queries, mobile devices [9] or content published in online social networks [2, 6] allows revealing further privacy sensitive data [14]. Among others, those include data on the users real name, address, gender, shopping-behavior or location [15, 4], personal interests, problems or desires of users, political or religious views, as well as the financial status.

Additional threats exist when attackers maliciously use advertising to distribute various forms of malware (malvertise) [107, 108, 109].

Search Engines:

All search queries made in popular search engines such as *Google.com*, *Yahoo.com* or *Bing.com*, can be saved and assigned to a user profile [110]. Is the provider able to connect the search queries to a specific user while being logged into the account, it is possible to enrich the users' tracking profiles. With the knowledge of the website content the user is interested in, the provider can personalize search results [92, 111, 112]. Moreover, through the identification of users, the provider can place suitable advertising in real time [91, 111], which is more profitable [94, 113, 114].

E-mail Provider:

E-mail providers have access to the entire e-mail communication, including recipient addresses and content. Evaluating this content can enable e-mail providers to gather personal data. Google Mail is an example processing the content of the users e-mails [91, 115], in order to provide topic specific advertising in the web-based interface.

[1] `http://acxiom.com/data-solutions/`
[2] `http://www.nielsen.com/us/en/nielsen-solutions/segmentation-strategy.html`

2.5.3 When Third-Party Tracking becomes First-Party

Tracking elements such as cookies are mostly embedded as third-party content, since all elements served from another domain than the visited website, are considered third-party. All elements served from the domain active in the opened browser window, are called first-party elements and are mostly not treated as tracking elements. Hence, it is possible to disable third-party cookies through browser settings or by using ad-block protection software to limit tracking cookies. Nevertheless, many tracking companies are able to set their cookies as first-party using various methods:

1. When logged into web services such as *Twitter*, *Facebook* or *Google*, the social media buttons *Twitter Follow*, *Facbooks Like* and *Googles +1* embedded on websites will report the visit and consequently the browsing history to the associated service.
2. Advertisers using pop-up windows or iFrames embedded on websites will set first-party tracking content through this new window.
3. Using HTTP redirects, a website will redirect users through the trackers website, set a first-party cookie, and redirect back to the original website, without the users notice.

2.5.4 Information Leakage

In a study [116] conducted in 2011 researchers exposed several methods in which first-party service providers leak user data to other third-party providers. This includes (1) exposing the e-mail address or user ID through the referrer header, (2) demographic information (gender, interests, ZIP code) in the request URI, (3) identifiers in shared cookies, (4) usernames or real names embedded in page titles. As an example, the following procedure is used:

```
1. GET http://ad.doubleclick.net/adj...
2. Referrer: http://submit.sports.com/...?email=jdoe@email.com
3. Cookie: id=35c192bcfe0000b1...
```

Here, the user is visiting the website *sports.com*. Because the website is embedding an ad provided from *doubleclick*, the browser sends a *GET* request to the *doublick.net* server (1). As an optional part of HTTP, the *sports.com* server sends within the referrer header the source URL and adding the logged-in user profile's e-mail address in the URL (2). As a response, *doubleclick* is setting a cookie with a unique identifier for this user ID (3).

Containing both, the e-mail address as part of the referrer, and the cookie identifier, *doubleclick* is able to link both information together and re-identify the user on further visits or other websites containing *doubleclick* ads. The researchers identified the leakage of data in 75% of the analyzed websites with this technique. In a similar study researchers confirm this behavior [12] and present further cases of leaked sensitive private data like e-mail addresses and real names to third-parties included as a parameter in the URL.

2.5.5 Buying User Data

The business model of many online companies is gathering and selling user data. For example, websites conducting user surveys like "Win new iPhone" have affiliations with tracking companies selling the conducted personal user data. In addition, these websites mostly embeds other third-party tracking elements. This makes it even easier to associate the user data with the browser history already conducted by the tracker. Advertising providers like *Datalogix.com* are buying user identifying information (name, ZIP code, address, ...) from marketing companies to create detailed user profiles and adjust targeted advertising [12].

2.5.6 Web Exploits

Third-parties use web exploits, e.g., cross-site-script vulnerabilities[3] on embedded websites to collect users' IDs or personal data [117]. Examples are: bugs in Firefox's error object [118], a bug in Google spreadsheets [119], using a technique called "history stealing" [120, 121], and bugs in Facebook Instant Personalization partner sites [122]. The mentioned bugs have been fixed by now, but as other exploits such as history timing [75] or clickjacking [123] attacks exist, and more exploits will be found and used in the future.

2.5.7 De-Anonymization of Users

Privacy risks of published user datasets are real, even when identifiers such as names or Social Security Numbers have been removed. Datasets can be de-anonymized by cross-correlating two datasets, like shown in early 1998 [124], when a discharge hospital database was joined with a public voter database. An early study [125] showed that only with the data of birth date, gender and ZIP code, it is possible to uniquely identify 87% of the US population. This leads to a huge privacy threat, whenever a web service requests detailed personal data, e.g., in the initial registration process. But even without real user data, a prominent case in 2006 demonstrates that users could be uniquely identified by analyzing publicly available search queries [36, 37]. This was possible after AOL published 20 million web search queries with the intention to help academic research. All queries were anonymized by assigning IDs to the searcher. By evaluating the search queries it was easy to identify specific users, like using this search query examples: "60 single men" and "dog that urinates on everything" and "landscapers in Lilburn, GA". These queries lead to a 62 year old widow who lives in Lilburn, GA., and loves her dogs.

In 2008 researchers [126] uniquely identified 99% of the user by applying a new algorithm on publicly available *Netflix.com* prize dataset, containing anonymous movie ratings of 500,000 subscribers. In contrast to previous work no assumption to a fixed set of *quasi-identifier*[4] attributes was made. The new de-anonimization algorithm used only a small amount of background knowledge about the user. Later, in 2009, the authors present a de-anonimization framework [128] for social networks and where able to re-identify 31% of *Twitter* users by using other social network data, in this case *Flickr*.

The browsing behavior data that online social networks gather, increases with the amount of their embedded social media buttons on the web. Many websites use these buttons to encourage interaction with the website and to improve their marketing. Visiting those websites having the social media buttons embedded, is sufficient to associate the content with the social media user account [116]. Further, experiments demonstrate that this data can also be assigned to real users without being logged into the social network [104]. This de-anonymization technique is very powerful to gain detailed personal user data. Marketing companies often share or sell "anonymized" survey data with tracking companies, enabling the re-identification of real users with the already gathered data by the tracker [117] or social media [129].

In an experiment [130] the authors showed that by crawling public groups in online social networks such as *Facebook* or *Xing*, and cross-referencing the crawled users with their browsing history, 37% of the users could be de-anonymized.

Another example is the de-anonymization of patient data from a German pharmacy data center [131]. Here the sold data was not sufficient anonymized and easily de-anonymized by the companies that bought the data for commercial purposes.

[3] `https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)`

[4] A quasi-identifier is a set of attributes that, in combination, can be linked with external information to reidentify (or reduce uncertainty about) all or some of the respondents to whom information refers [127].

2.5.8 Price Discrimination

Many people might not see it as a threat, if services like *Google* or *Amazon* display personalized advertising, since no harm is done by displaying ads. But people will agree that by using this data companies can modify the displayed content, like the price of a product, will affect everybody.

Researchers found out [20] that depending on the geographic location or the website the user visited before, the price of a displayed product differs up to 166%. Another study [132] by the *Wall Street Journal* confirms that websites such as *Staples Inc.* displays different prices to users in different locations. *Staples* showed a lower price when a rival store was near to the user's geographical location. Within this investigation several companies reported to the journalists that their online price modification mirrors the real world. Local stores usually adjust their prices to account for local demand, competition or store location. But the examination of *Staples'* online pricing also discovered differences depending on the weighted average income among ZIP Codes, based on Internal Revenue Service data.

Further, the travel agency *Orbitz Worldwide Inc.* [133] found out, that customers using an Apple Mac tend to spend 30% more on hotel bookings on their website then Microsoft Windows user. As a result, they started to advertise more expensive products for users browsing with a Mac. Similar, researchers found evidence [134] that the type of device (mobile or desktop PC) as well as the browsing history influenced the price of popular websites such as *Home Depot*, *Hotels.com* or *Expedia*. On *Expedia* and *Hotels.com* a technique marketers call "A/B tests" was used to lead a subset of the users towards more expensive offers. Depending on the users' cookies (browser history), the companies divide the users into different groups A/B/C and display hotels with an average price of $187 for group A and B or $170 a night for group C. *Home Depot* on the other hand displayed results on an average price per item of $120 for desktop users, and $230 for mobile users. These examples demonstrate that price discrimination is present and could be applied according to many other personal data like race, gender or financial background.

2.5.9 Financial Credibility

Tech startups like *Lenddo*, *Kreditech* or *Kabbage* are companies determining a users' financial creditability on data based on *Facebook* friends, *eBay*, *Amazon* or *PayPal* accounts [16] before lending money. For example, *Lenddo* sets the user's creditability depending on weather the *Facebook* friends are late paying back a loan to *Lenddo*. Users borrowing money from *Kabbage* even need to grant access to their *eBay* and *PayPal* account. *Kabbage* states that it can determine the creditworthiness and put money into the user's account in just seven minutes. For this, *Kreditech* is using account data gather from *Facebook*, *eBay* and *Amazon*.

In Germany, the largest credit ranking company *Schufa* planed to mine *Facebook* and other social networking sites to rate person's creditworthiness [135]. According to *Schufa*, the goal was "identifying and assessing the prospects and threats". Due to intense public and political critics the idea was never realized.

2.5.10 Insurance Coverage

Tracking can have a negative affect on insurance rates of customers. Consumer habits derived from credit card spendings, magazine subscriptions or product surveys as well as the online browsing behavior reveal a lot about the interests and the lifestyle of users. This data can be used by insurance companies to estimate the risk of accidents through the users hobbies or getting ill because of an unhealthy life [136]. Through consumer-marketing data, traditionally used for advertising purposes, insurance companies can for example estimate a person's risk for illnesses like high blood pressure or depression. This data is supported by algorithms based on *Deloitte's* model [137] assuming that many diseases correlate with factors such as fast-food diets and low exercise habits.

Acxiom Corporation, one of the biggest data gathering companies, uses 3 billion data items daily. Those include detailed online purchases as well as browsing history and connects this data with personal data collected from online social networks. This data helps to identify potential actions for insurers on different customers. The largest marketing-database companies in the U.S. including *Acxiom*, *Alliance Data Systems Corp.*, *Experian*, *PLC*, and *Infogroup*, claim to have detailed information on more than 100 million U.S. households [137].

2.5.11 Government Surveillance

In the wake of the PRISM [138] scandal in mid 2013, the public was made aware of the extent of Internet surveillance by intelligence services. Here the Tempora [139] and Upstream [140] projects are able to spy on a large percentage of Internet traffic and share their findings.

Because of the security features of SSL/TLS [141], the are limits to which data can be evaluated by the governmental surveillance agencies. Therefore, the data that tracking companies obtain on users is valuable for government agencies and law enforcement authorities. In 2015, governments of all countries made 35.365 requests for 68.908 user data on *Google* [142]. This includes personal data such as name or address as well as all data obtained by *Googles'* products such as service usage, geolocation, *Google Mail* content, search history, telephone calls through *Google Voice*, *Youtube* videos or chats using *Hangout*.

By collecting vast amounts of Internet traffic, including the HTTP meta data and cookies [143], the agency can analyze the users online activities. Since tracking user behavior is done by unique identifiers in cookies, governmental surveillance agencies like the *NSA* and British *GCHQ* make also use of this technique [144] evaluating the collected Internet traffic data. According to the documents provided by Edward Snowden [144], both agencies use *Googles* specific "PREF" cookie, containing an unique client identifier. This cookie is usually set by all *Google* services on the web.

A document provided by *The Guardian* [145] revealed, that the *NSA* and *GCHQ* also evaluated *doubleclick* tracking cookies, in order to identify users browsing with the Tor browser (see Section 2.7.8) and switching to normal browsing. Because many Tor browser users seam to not deleting cookies while browsing with or without the Tor browser, their identity can be easily exposed by matching the cookies during both browsing sessions.

2.5.12 Positive Aspects of Tracking

There exist also benefits of tracking and the exposure of private data. A study by Beales [146] surveying 12 ad networks in 2009 showed that behavioral targeting increased the percentage of users clicking on an advert by a factor of 7.7 and more than doubled the conversion rate in comparison to non-targeted advertising. The study shows that users do find targeted advertising more relevant than completely random ads. On the other side, not only the advertising companies benefit from making money, the website owners profit too, being able to provide the services for the users.

2.6 Web Tracking Techniques

Methods and techniques to track users on the web evolve over time. While there are different forms of tracking, we divide them in groups: session based, storage based, cache based, fingerprinting and others.

2.6.1 Session Based

IP Address:

The simplest tracking form is based on collecting IP addresses. Every device connected to the Internet has a valid IP address. However, this form of tracking has its drawbacks. Even if computers need to have its own IP address to receive data, there exist several cases where the unique user IP address is hidden: (1) Behind a NAT or proxy in a private or company network. (2) Using a Virtual Private Network (VPN) or anonymity networks like TOR. (3) Frequently changing IPs on dial-up connections [147]. For those reasons IP addresses are not sufficient for tracking users.

Session Identifiers:

Before cookies were introduced in 1994, web services as well as trackers started to pass session identifiers to another website in the URL (GET method) or as a value in a (hidden) field of a web form (POST method) [148]. This identifier can be any string being able to uniquely identify a user during a single browsing session. Although these technique can still be used, they are only limited to one session and one browser [147]. However, the identifier could internally be passed to a third-party or (nowadays) saved in a cookie.

Document Object Model:

The Document Object Model (DOM) provides another form of tracking [149]. With its access to the content of web documents, it is also possible to store data in the websites document objects (e.g., *window.name*). Combining multiple values encoded in JSON strings in the *window.name* property, it will be resistant to page reloads and is accessible from other domains as well. This gives third-parties the opportunity to exchange user data without cookies.

2.6.2 Storage Based

HTTP Cookies:

Storing and retrieving persistent data is an efficient method to track users. As our analysis [23, 22] shows, HTTP cookies[5] are the by far mostly used tracking method. This still holds, even when browser vendors nowadays implement options to delete cookies. This popularity is because cookies can be set automatically on any HTTP request, store up to 4KB data, with minor bandwidth usage and without any user interaction or user notice. A tracking pixel (also known as web bug or web beacon), a transparent 1×1 pixel sized image, is the traditional, bandwidth-conserving, and high-performance way of doing it. The steps that happen on the HTTP level are shown in Listing 2.1. Once implemented, the visitor's browser will request the tracking cookie each time a page is loaded (assuming no caching), getting and transmitting a (third party) cookie. A website can also set so called *session cookies* used for example for web-authentication, but because of a limited lifetime the browser will delete those cookies automatically after closing.

While cookies are used for other purposes such as storing login session data, shopping carts or user specific website settings, the challenge is to correctly identify tracking cookies. To actually perform tracking, the trackers have to be able to re-identify users. They can either use static user data directly,

[5] `http://tools.ietf.org/html/rfc6265`

```
# first request
GET /pixel HTTP/1.1
Host: tracker.com
Referer: http://www.some-site.com/

# first response
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Length: 43
Last-Modified: Thu, 08 Aug 2013 22:07:11 GMT
Date: Fri, 09 Aug 2013 14:03:08 GMT
Set-Cookie: id=74abd0e32dbec5db124ff1aa97135bde; Expires=Tue, 19 Jan 2038 3:14:07 ↩
GMT

# following requests
GET /pixel HTTP/1.1
Host: tracker.com
If-Modified-Since: Thu, 08 Aug 2013 22:07:11 GMT
Referer: http://www.some-site.com/product/42
Cookie: id=74abd0e32dbec5db124ff1aa97135bde

# following responses
HTTP/1.1 304 Not Modified
```

Listing 2.1: Tracking pixels on the HTTP level. The server defines a third-party cookie on the first request that gets sent back by the browses on each subsequent request. The private information (the visited site) is leaked via the `Referer` header.

such as the IP address or browser fingerprint, or they have to store a custom identifier and retrieve it later.

In comparison, non-tracking or session cookies have a fixed expiration time limited to the browser session or one day. Using this findings the researchers were able to identify tracking cookies by a precision of 99.4% and recall of 100%.

Nowadays, browser vendors try to prevent tracking by disabling third-party cookies by default. In addition, there exist many browser extensions such as Ghostery[6] or AdblockPlus[7] and various research projects trying to detect and prevent cookie tracking [150, 12, 151, 11]. But as our research in Chapter 4 shows, trackers can still set their cookies by redirecting the user to the trackers website, setting a first-party cookie, and redirecting them back. Similar behavior is possible [13] using pop-up windows or iFrames, where the tracker will set it is first-party cookies on the users machine. Unfortunately, even if the awareness of tracking rises, research confirms that only 30% of users delete their cookies within a month from their acquisition [152].

Cookie Syncing:

Another privacy threat for users exists through cookie syncing between companies [13, 3]. Companies that are storing a cookie by visiting a domain (e.g., *bing.com*), pass their cookies also to other domains (e.g. *msn.com* and *live.com*), without the user ever visiting those [12, 56]. The exchange of user information and identifiers enables companies such as Microsoft and *Google* to facilitate targeted ads and real time advertising across many websites [12, 153].

[6] `https://www.ghostery.com`
[7] `https://adblockplus.org`

Advertising Networks:
Advertising networks like *admeld.com* cooperate with other trackers to share aggregated user information [13]. Through the embedded ads on websites, advertising networks make further requests to cooperating trackers inside the ad code. The request includes a unique user id and the embedded website URL. This makes it possible for other trackers to track the user across multiple sites, without the need to set their own cookie.

Adobe Flash Cookies:
Similar to HTTP cookies, Adobe Flash uses Local Shared Objects (LSO) and Remote Shared Objects (RSO) to store persistent tracking data on the users' computers [154]. Those objects can store up to 100KB of data, are centrally accessible through different browsers, do not expire by default, and are even harder to delete [44, 155]. These properties enable a greater tracking potential compared to HTTP cookies. Additionally, Flash can make us of the LocalConnection object[8] to communicate between Flash instances such as SWF files and Flash cookies in normal and even in private browsing mode at the same time [53].

Microsoft Silverlight and Java:
The competitor Microsoft supports with its product Silverlight a similar Isolated Storage object to track users [38]. Using the JNLP Persistence-Service [156], also Java provides a method to store tracking data on the users' computer. Because of the continuing decreasing usages of those features, the tracking capabilities are limited.

HTML5 localStorage:
HTML5 introduced in 2011 a new web storage which was quickly integrated by all browser vendors and similar used by tracking companies [157, 155]. This browser built in feature is capable to store up to 5MB data objects per site, until it is deleted by the user or the website, giving it an advantage over cookies. Although the usage of the HTML5 localStorage feature for tracking seam to be low at the moment [44, 53] comparing to HTTP cookies, it can be used to reconstruct other deleted tracking elements [158].

Internet Exporer userData Storage:
A specific Internet Explorer tracking method is possible through the feature *userData* storage [12]. UserData can store persistent information across sessions. Capable of storing 128KB data it behaves similar as HTML5 Local Storage. The data structure is more dynamic and has a greater capacity than cookies[9]. This feature is not integrated in the newest browser Microsoft Edge.

Web SQL Database:
Web SQL Database[10] (which later became HTML5) IndexedDB[11] gives another possibility to track users [57] within built-in browser features. This database storage is comparable to HTML5 Local Storage, and can be used to save and reconstruct tracking cookies.

Server Log Files:
Visiting a website, the browser connects to a server for downloading the websites' content. In this process, the server will store this data, e.g., the IP address, user agent and the requested pages in the servers log files. As long as this data stays on the server and is not shared with other companies, this kind of tracking does not untermine the users' privacy.

[8] http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/flash/net/LocalConnection.html
[9] https://msdn.microsoft.com/en-us/library/ms531424(v=vs.85).aspx
[10] http://dev.w3.org/html5/webdatabase
[11] http://www.w3.org/TR/IndexedDB

2.6.3 Cache Based Tracking

A common practice of tracking companies is to identify a user by obtaining the web browsing history [44]. This is possible with different cache based techniques. In general, trackers would include a specific HTML file in the websites' source code, containing a unique identifier. Once the user visits a website, an included tracking file will be stored in the browser's cache. Then, the tracker can determine which websites have been visited by checking the cache's tracking file.

Embedded Content Timing Attack:

Websites can use JavaScript to measure the time it takes to display content on websites. Thereby the script is determining, whether the content such as an image that is placed on a website is downloaded from the server, or pulled from the browsers cache. When the time to display the images is short, it is an indication that the image was in the browser cache and the user already visited the website. This timing attack can be used to check many images, for example the company logos of various websites, to reconstruct the browsers history of visited websites [159]. Similarly, JavaScript can indirectly cause a DNS lookup and evaluate the time needed. In case the website has been accessed before, the corresponding entry will exist in the DNS cache and reduces the lookup time significantly.

HTTP Header:

The HTTP header provides the following fields on the first download of an object: *Last-Modified*, *ETag*, *Cache-Control*, and *Expires*. With 81864 bits the *ETag* (entity Tag) field can be used by trackers to store a unique identifier [12]. Further, the *Last-Modified* header can be contain any tracking string (unique user tracking identifier) and not only a date [44]. In this process, when a user re-visits a website, the browser sends the *If-Modified-Since* and *If-None-Match* headers as a part of the HTTP request. These headers contain values stored in the *Last-Modified* and *ETag* fields from the previously cached document. Subsequently, the web server checks if the cached copy's time stamp is valid. If the cached copy is still valid the server returns a short response with the *HTTP 304 Not Modified* status. In case it is outdated a new document is returned [53].

Redirect Cache:

The HTTP status code 301 can be used for tracking in the following setup: A user visits a website containing a hidden tracking iFrame. The iFrame redirects the user to an URL containing an unique identifier. This identifying request is cached and used by the browser during the next attempts to access the website [160].

Authentication Cache:

Grossman [161] showed a method to track users by using the HTTP authentication cache. In doing so, a JavaScript forces the browser to authenticate to the web server without presenting an authentication request to the user. Once the browser is authenticated, and after receiving the HTTP status code 401, the browser will cache the credentials and send it with each subsequent HTTP request.

TLS Sessions:

C tracking technique [162] makes it possible to check if a website was already visited by using the TLS session resumption cache and TLS/SSL session IDs. The TLS session resumption cache stores TLS/SSL session IDs and avoids full TLS handshakes by reducing latencies. Those are sent by the server to the client during the *hello message* in order to use them when connecting to the host in the future. When a tracker checks for a list of websites if the session IDs are already present in the cache, it can reconstruct the users browser history.

2.6.4 Browser Fingerprinting

Browser fingerprinting refers to a group of techniques to gather all possible device properties in order to uniquely (re-)identify a user without storing a custom ID and without the users notice. Fingerprinting scripts embedded on websites use a broad range of technologies such as JavaScript, Flash, HTML5, CSS, ActiveX or HTTP and TCP headers [53, 55] to retrieving those properties. This unique user identification originates from various hardware, network, or software configurations that a user's device may have [45, 49, 52, 53]. This information, also called fingerprinting features, are mainly used by companies to track user behavior on the web [30] and to establish detailed user profiles [24, 35].

Because tracking is happening in the background, users have no means to know if being tracked. Since this tracking technique is in-transparent and relatively new, most users are not aware of it, and therefore not using any protection methods against it. In addition, as our research in Chapter 6 shows, the protection methods available have various flaws and are not able to protect the user properly against fingerprinting.

The following list summaries various fingerprinting features used by trackers we gathered in our research:

System information: Device ID, operating system (version, architecture, kernel), screen resolution, screen height, screen width, color depth, pixel depth, timezone, system fonts, system language, date and time, CPU type, clock skew, battery status, mouse movement, keyboard layout, accelerometer, multitouch capability, microphone, camera, audio capabilities, printer support.

Browser information: browser version, browser vendor, User Agent, navigator object, installed plugins, preferred and accepted languages, accepted HTTP headers, cookies enabled, supported MIME types, browser history, do-not-track, HTML canvas element, JavaScript runtime, CSS Features (font probing, unicode glyphs, display, position, u.a.).

Network information: IP address, Geographic location, TCP timestamps, TCP/IP parameters, proxy-piercing.

Flash information: version, manufacturer, serverString, language, screenDPI, screenResolutionX, screenResolutionY, getTimezoneOffset, getLocal, XMLSocket, Math.min, Math.max, ExternalInterface.call, ExternalInterface.addCallback, sendAndLoad, URLLoader, navigateToURL, loadMovie, createUID, getUrl, allowDomain, allowInsecureDomain, loadPolicyFile, URLRequest, LoadVars.

Network Fingerprinting:

Network information can easily be determined by a tracking element embedded on a website and checked through a HTTP request made to the server. By using the IP address the tracker can calculate the geo-location. If a computer is inside a local network or proxy, the tracker can use a Flash script to bypass the proxies in order to discover the real IP address of users [30].

Even when web browsing privacy mechanisms, such as Tor or encrypted tunnels (IPSec, SSL, VPN) are in place to hide the content of the data transferred, web page fingerprinting is possible. Research has shown that these defenses do not obscure the size, direction, and timing of packets transmitted between clients and remote servers exposing the visited website [60, 163].

Proposed defenses, such as traffic morphing [164], randomized pipelining over Tor [165], splitting individual HTTP requests into multiple partial requests (HTTPOS) [166] try to stop these attacks. As Cai et al. [76] shows, also these defenses have flaws exposing the visited website. The race between attacks and defenses continued leaving an open question in a practical deployment, since it requires substantial changes of Tor or the TCP stack [167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178].

System and Browser Fingerprinting:

Researchers showed that using JavaScript and Flash scripts fingerprinters can identify very detailed system properties [51, 179, 50, 180, 181, 30, 46, 57, 182] such as system fonts or the operating system architecture. Further, very robust browser features can be retrieved using CSS, JavaScript and HTML5 such as unique graphic card rendering characteristics [49], JavaScript engine or performance [52, 49, 45] or CSS font probing. Researchers [49] identified in HTML5 242 tags, attributes and features in HTML5 that were suitable for browser identification. A detailed discussion of browser fingerprinting can be found in Chapter 6.

Smart Phone and Search Query Fingerprinting:

In addition to the browser, more techniques exist to uniquely fingerprint users on smart phones [8]. Kurtz et al. [183, 184] presented a feature list to identify Apple iOS users through their personalized device configurations and applications. Other researchers use different smart phone sensors like the camera, microphone, speaker or accelerometers to fingerprint the device [185, 186, 187, 188, 189].

Website fingerprinting can also be extended to fingerprint individual search queries or keywords to web applications [10]. By augmenting traffic analysis using a two-stage approach with new task-specific feature sets, a passive network adversary can in many cases even defeat the use of Tor to protect search engine queries.

2.6.5 Redirect Tracking

Redirects can be used in various scenarios and are mostly utilized to enhance the usability of the web. For example if a website is either too slow, not valid anymore, or when the content is changing often, the website can redirect the user to another, valid, new or updated web page. The common use case is when a URL has moved temporarily or permanently to a new URL. The web server will then send the HTTP status code 302/303/307 (moved temporarily) or 301/308 (moved permanent). The defined differences between the two groups of redirect status codes are the cache ability and the request method – see details in RFC7231[12]. It is also common to use client side redirects with HTTP meta refresh, JavaScript or PHP, see the example code in Listing 2.2.

Nowadays, redirects are used for URL shortening or privacy protection by hiding the HTTP referrer when leaving the website. On the downside, redirects are also being misused for phishing attacks [190, 191, 192, 193] or to deliver malware through drive-by-download attacks [194, 195, 196].

Since several privacy tools and add-ons try to protect web users' privacy by blocking third-party connections, advertisers use more and more advanced techniques like redirects to place tracking cookies.

As part of a website, redirect link tracking starts with clicking basic text links as depicted in Figure 2.2. Redirect links can also be set on image or dynamically generated links via JavaScript that suppose to lead to further content, but actually detour the user over a chain of other (tracking) servers first. While the appearance of those (redirect) links is exactly the same as other legitimate website links, the user will not notice the difference and can not prevent the action when following the links.

As seen in Figure 2.2, the website *A.com* embeds *Link 1* and *Link 2* inside their website's content/navigation. By clicking the links, both navigate the user to further content or services like the *News* or *Jobs* page of the website *A.com*.

When clicking *Link 1*, the website (server) will directly open the *News* page of *A.com*. If the user clicks *Link 2*, the server will first connect to the tracking website *B.com*, then, automatically redirect to another tracking server (*C.com*) and finally redirect to the actual *Jobs* page of *A.com*. The redirection through a chain of potential tracking servers makes it possible to store various tracking cookies. In general, those redirect tracking websites have no content that is rendered (displayed) in the browser. Loading those connections takes only a few milliseconds, so a normal user will neither notice nor be able to prevent that

[12] `https://tools.ietf.org/html/rfc7231`

```
HTML meta refresh:
<meta http-equiv="refresh" content="1; URL=http://www.x.com/redirect.html">

JavaScript:
<script>
  document.location.replace("http://www.x.com");
  window.location.assign("http://www.x.com");
  window.location.href = "http://www.x.com";
  window.location = "http://www.x.com";
</script>

PHP:
<?php
  header("HTTP/1.1 301 Moved Permanently");
  header("Location: http://www.x.com");
  header("Connection: close");
?>
```

Listing 2.2: Client side redirect code using HTML meta refresh, JavaScript and PHP

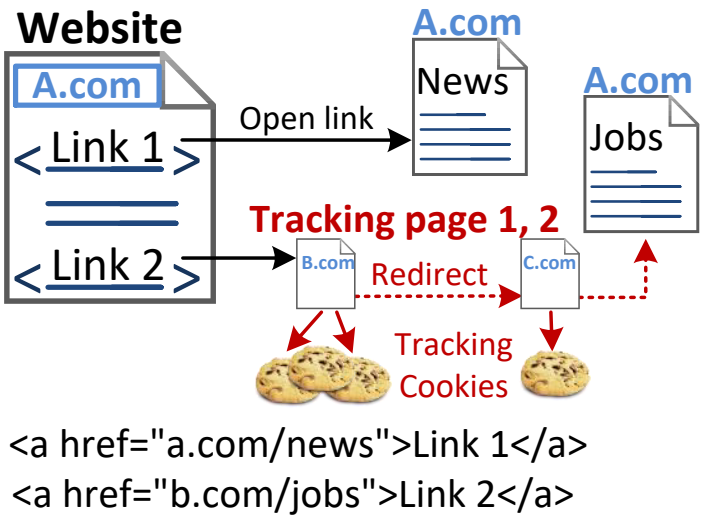

Figure 2.2: Redirect link tracking behavior

other websites are being loaded. During the process of loading the tracker websites, the server can collect different meta data and use various tracking techniques to be stored on the user's machine. Because the tracking server is being accessed directly (by clicking the link and opening a new connection), first-party tracking cookies can be stored which is not going to be blocked by anti-tracking tools, as these tools mostly block third party elements. The tracking information stored inside the cookies can contain data such as the visited website (*A.com*) and what element/content was clicked (*Link 2 / Jobs*). The result is a much richer user profile compared to basic tracking techniques. There are not many indications for users to recognize redirect links since they appear and behave like ordinary links. Even checking the status bar (bottom right corner of the browser) of each link can be misguiding since JavaScript could dynamically update the link. A real redirect scenario is presented in Figure 2.3. Here, clicking the link *Jobsearch* on the the website *www.spiegel.de*, will open for a few milliseconds different websites in the background of

the browser using automatic redirects. In addition, tracking cookies are saved on the users' machine. The final redirect will open the (legitimate/user intended Jobportal) *stellensuche.karriere.spiegel.de* website.

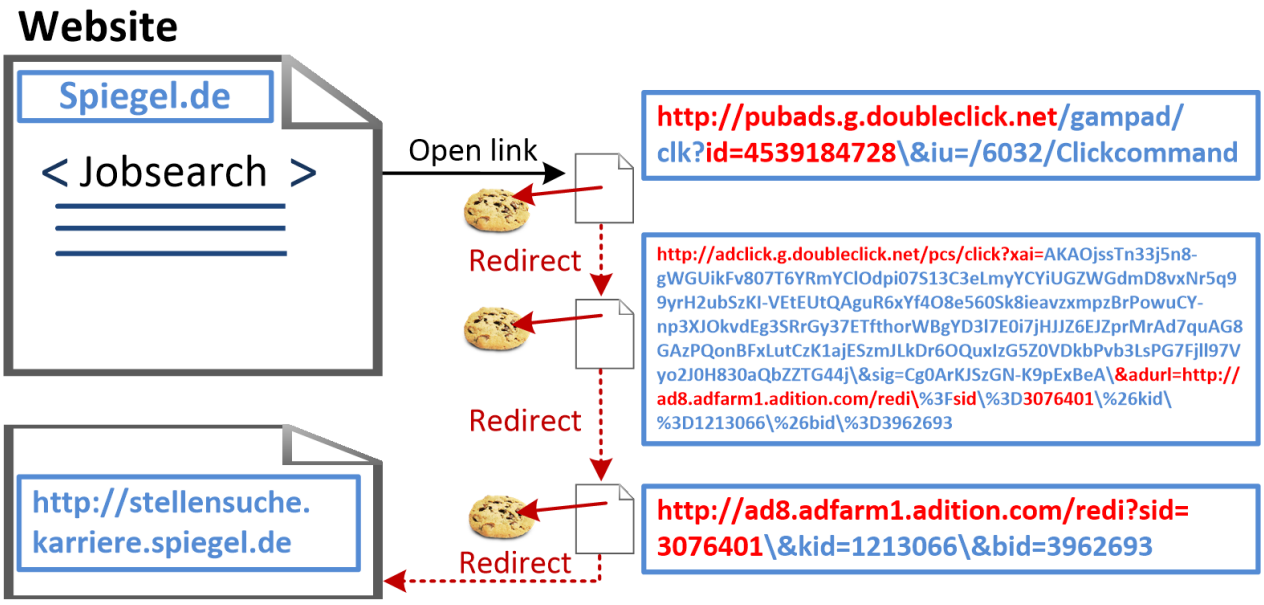

Figure 2.3: Real redirect example when clicking the 'Jobsearch' link at the spiegel.de website

2.6.6 Others

HTTP referrer:

Another way to transmit tracking information about visited websites is through the HTTP referrer header. The referrer header is attached to outgoing HTTP requests (while following a link) and can be set optionally by the server to add the current URL. In addition, a tracker could use a JavaScript code to directly transmit the URL by accessing the *document.referrer* through the API or send the information through the GET / POST parameter of a request to the tracker's domain [13]. *Verizon Wireless* for example injects a special X-UIDH header to every outgoing HTTP request with a temporary ID [197]. Websites can then track those users and buy personal information directly from *Verizon*.

Ultrasound Cross-Device Tracking:

This method enables user tracking across multiple devices, such as smartphones, television and computers [198]. Using inaudible sounds (ultrasound) emitted by one device and detected by the microphone of another device only built-in hardware is necessary. Those signals are also called "audio beacons" and can be embedded into TV commercials or played by browser ads and be detected by mobile apps running in the background. While the sound can not be heard by the human ear nearby devices can detect it without extra hardware. Then browser cookies can pair a user to multiple devices and keep track of what TV programs the users watch or how he interacts with the online ads. Researchers discovered 234 Android apps that ask permission to access the smartphone's microphone in order to incorporate a particular type ultrasonic beacon [198]. In addition, the researchers found that 4 of the 35 retail stores they visited in Germany have ultrasonic beacons installed at the entrance. This method is already in use since 2015 [199].

Combination and recreation:

Yet another sophisticated tracking method combines various tracking techniques. As discussed earlier, users can delete cookies and prevent tracking. Therefore trackers started to use Flash or JavaScripts

to duplicate and reconstruct deleted tracking cookies [57, 200]. Later, researcher also presented that ETags and HTML5 localStorage were used to rebuild HTTP cookies [44]. Here, the researchers found that a popular advertising company *KISSmetrics* was using this technique to track users on the popular tv streaming site *hulu.com*.

Within the *evercookie project* [158] research demonstrated how to reconstruct a extended list of different tracking elements including: HTTP cookies, Flash Local shared Objects, Sliverlight Isolated Storage, storing cookies in web history, HTTP ETags, storing cookies in web cache, caching in HTTP Authentication, *window.name* caching, Internet Explorer userData storage, HTML5 Session Storage, HTML5 localStorage, HTML5 Global Storage, HTML5 Database Storage via SQLite, Java JNLP PersistenceService and even storing cookies in RGB values of auto generating, force-cached PNG's using HTML5 Canvas tag to read pixels (Cookies) back out.

The Tor browser shows other potential tracking options and which are covered by the Tor browser [201]: HTTP auth, SSL state, OCSP state, site-specific content preferences (including HSTS state), searchbox and findbox text, the Google WiFi geolocation token and the safe browsing key.

2.7 Tracking Protection Mechanisms and Tools

The market is reacting on the user's disapproval of online advertising. As research shows [202, 203, 58], 78% of users do not want to see ads on websites. Users are concerned about their loss of privacy through web tracking which can be seen in the exponential rise ad-blocking tools [204]. From 2010 to 2015 the usage of ad-blocking tools increased from 21 to 198 million. Including mobile devices, the market has grown to 615 million ad-blocking devices in 2016 [205].

Over the years, a variety of approaches towards protection of data privacy have been proposed, ranging from blocking a specific tracking technique to hiding the users information. This section evaluates the solutions to detect and prevent web tracking. A detailed discussion on protection mechanisms against browser fingerprinting is provided in Section 6.1.

2.7.1 Clearing Browser Cookies, Cache and History

Although it is a good strategy to impede tracking by deleting all browser/Flash cookies, clear the cache and history manually within the browser, it has its challenges and drawbacks. Most users are unaware of these options or do not have the technical knowledge to accomplish this task [38]. Even when users delete cookies, they forget to clear tracking elements such as Flash cookies, the cache and history as well. There exist tracking techniques that can re-create deleted cookies, as discussed in Section 2.6.6. Users need to clear all browser data regular to prevent tracking effectively. To delete Flash cookies, users need to remember to do it in the Flash Player properties.

As an alternative, users can use tools such as Click & Clean[13] or BetterPrivacy[14] to clear the browser's cookies, cache, history, and more tracking elements. The problem with those tool is their limited tracking protection on selected tracking techniques and the rare usage. If the user is not executing the tools regularly, tracking is still possible.

2.7.2 Built-In Browser Mechanisms

Although modern web browsers allow to disable third-party cookies, this method does not protect the user from tracking due to two factors. First, the handling of disabling third-party cookies differs in their implementation in browsers. When third-party cookies are disabled, Google Chrome for example, only

[13] `http://www.hotcleaner.com/clickclean_chrome.html`
[14] `https://addons.mozilla.org/de/firefox/addon/betterprivacy`

prohibits sending third-party cookies back to the tracker, while Firefox prohibits storing and sending third-party cookies. Second, this implies that all cookies are present in the user's browser and new first-party cookies that are set by trackers through other methods such as pop-ups, iFrames or JavaScript, will still send tracking data [13].

Microsoft was the first browser vendor to implement a feature that was able to block third-party tracking back in 2011 [206]. The so called *Tracking Protection List* (TPL) was available for the Internet Explorer 9-11 and needed to be set up by the user with one or more lists filtering tracking domains (not provided by Microsoft). In cooperation with Microsoft Germany we implemented an automated system to create such a list of tracking domains and made it available for public. The work is presented in Chapter 3.

In 2016, Mozilla Firefox (version 42) enabled ad-blocking by default [207] in the private-browsing mode (see Section 2.7.3). In cooperation with the tracking protection browser add-on *Disconnect*[15], Firefox uses the provided tracking list to block tracking content (including ads, analytic trackers, or social media buttons such as the Facebook Like button) [208]. After three years of evaluation, Mozilla finally decided to activate this feature in the default browser mode starting Version 65 beginning 2019 [209].

Similarly, the Opera browser added native ad-blocking in 2016 [210] using the EasyList[16] tracking filter list. With those mechanisms both browsers are able to block content loaded from domains using predefined URL filter lists in order to stop tracking users across websites.

Apple followed in 2015 and introduced an add-on to manipulate page resources in the Safari browser. This made it possible for ad-block software to integrate into Safari [211]. With the rising popularity of tracking protection, Apple announced in 2017 its native *Intelligent Tracking Prevention* (TP 1.0) [212]. Here, *WebKit*[17] has extended its features to reduce tracking. With blocking third-party cookies by default, ITP 1.0 reduces cross-site tracking by limiting 3rd party cookies and other website data to be accessed only for one day and being purged after 30 days. Early 2018 Apple updated the tracking protection to ITP 1.1 [213] by moving 3rd party tracking cookies in a partitioned space when the domain setting those cookies was not visited within 24 hours. Trackers were not able to access those cookies or set new onces. The cookies were deleted after the domain had not been accessed within 30 days. Because people complained that 3rd parties were still able to track the user within the 24 hours limit, Apple introduced ITP 2.0 in summer 2018 [214]. Similar to the proposed cookie partinioning, described in Chapter 5, Apple removed the 24-hour cookie access window to immediately partition cookies for domains determined to have tracking abilities. With ITP 2.0 in place, websites now have to request tracking privileges and users must specifically accept. Furthermore, two additional features are implemented in Apples tracking detection: *Protection Against First Party Bounce Trackers* and *Protection Against Tracker Collusion*. The first feature is going to detect trackers that have not been used as a third party content provider but tracks the user through navigational redirects. The technique has been discovered in 2014 (see Section 2.6.5). Chapter 4 provides further details. The tracker collusion detection is described in research as cookie syndication, see Section 2.6.2. Moreover, with restricting the browser to only load built-in fonts, not providing detailed browser plug-in information and sharing just general system settings, Apple is following the similar strategy on limiting browser fingerprinting as we proposed in Section 6.2.1 (many browsers, one configuration). This will make a Mac look more like everyone else's Mac. How effective those features are is yet unclear leaving it a future work for research.

[15] `https://disconnect.me`

[16] The EasyList filter lists are sets of rules originally designed for the browser add-on 'Adblock' to automatically remove unwanted content such like 3rd party tracking, ads, or other tracking scripts from the Internet. URL: https://easylist.to/

[17] WebKit is the web browser engine used by Safari and many other apps on macOS, iOS, and Linux.

2.7.3 Private Browsing

In 2008, browser vendors implemented a privacy oriented browsing mode known as "Private Browsing" (or "Incognito", as used by Google Chrome). In an ordinary browsing session, all stored cookie, cache or history elements are kept until they are erased manually. Effectively, the private mode behaves like a separate browser profile. After the private browsing session is terminated, all saved items such as cookies, cache and history are automatically deleted. The advantage is that tracking elements are deleted automatically. However, this method has the drawback that tracking is still possible during the entire private browsing session. Even if the data is deleted after the browsing session, the tracker can still store cookies and evaluate the browsing behavior within the session [157].

2.7.4 Opt-Out Cookies

Over the years the advertising market noticed the users' disprove towards ads. On this behalf, many advertising companies unite in associations such as the Network Advertising Initiative (NAI)[18], Digital Advertising Alliance (DAA)[19], Interactive Advertising Bureau (IAB)[20] or European Advertising Standards Alliance (EASA)[21]. These associations represent the customer's interests as well as regulatory demands by governments towards advertising companies.

In order to protect the users rights the associations developed a method to "opt-out" of personalized cookie tracking [215]. Personalized tracking means the usage of the tracking data gathered through the users browsing activities. For example, if the user is searching for new sport shoes, visits different websites providing sport shoes, a tracker can make use of the online behavior to display personalized advertising of sport shoes. Trough websites like *http://www.networkadvertising.org/choices* the user can select all those participating advertising companies to opt-out of personalized advertising. A script will then set specific cookies for every selected advertising company in the users browser. Within this cookie a code indicates to the advertiser that the user disapproves the collection of personal information and does not want to receive personalized advertising. As stated in the regulations [215] the collected information should be anonymized. However, the advertiser will still be able to serve ads to the users and and use web tracking cookies to collect user information. Currently, there is no regulation on the type of collected data, so the user has no knowledge on the mode of operation. In addition, it does not cover the use of other web tracking technologies that *NAI* member companies may use to deliver personalized advertising.

According to a study [216], the usage of opt-out cookies is around 1%. This can be explained by the minor knowledge about this technique but also because of their associated disadvantages [12]: (1) Opt-out cookies does not cover all advertising companies. (2) Web tracking and personalized advertising is not blocked completely. (3) Opt-out cookies have a limited lifetime. (4) Users needs to set and maintain opt-out cookies manually. (5) If the user deletes the browser cookies, opt-out cookies will also be deleted. (6) For using opt-out cookies, user needs to activate third-party cookies in the browser which will allow the storage of other tracking cookies automatically.

2.7.5 Do-Not-Track Header

Researchers [217] developed a HTTP mechanism that should solve the drawbacks of opt-out cookies. The project was picked up and endorsed by the US Federal Trade Commission (FTC)[22] in their privacy report

[18] `http://www.networkadvertising.org`
[19] `http://www.aboutads.info`
[20] `http://www.iab.net`
[21] `http://www.easa-alliance.org`
[22] `https://www.ftc.gov/`

[218]. The W3C worked on a standardization [219, 220] and soon after browser vendors implemented the mechanism as "DNT" (Do-Not-Track) header [221], an initiative also supported by the European commission [222].

If the header is set to $DNT : 1$ advertising companies have to respect the users' wish to stop behavior tracking. This mechanism can be seen as a built-in "opt-out" cookie serving the same purpose. But still, not all browser implement the feature in a similar way. For example, Mozilla Firefox and Microsoft Internet Explorer set the DNT header by default to 1, Google Chrome sets the header by default to 0 (disabled). Furthermore, there exists neither an standardization nor enforcement on how an advertising company must response to the received DNT header. Some companies may replace the unique identifier in their tracking cookies with a generalized ID, preventing user identification. In this case, the companies can continue collecting the same information but all users would be aggregated. Another advertising company may respond by by maintaining their method of data collection but alter its usage. Although DNT should prevent behavior advertising, users' still have no transparency on how their data is collected and used.

2.7.6 European Cookie Law

In 2009, the ePricacy directive [223] on the legislation on cookies to regulate the usage of behavioral cookies was extended. As stated in Article 5(3), website owners are required to inform the user prior to the storage of cookies and access to information stored on a user's machine. In other words, all websites which embed tracking elements such as third-party cookies, need to inform the user in an appropriate manner about the usage of cookies.

The ePrivacy directive needs to be enforced by legislation of all European countries but does not regulate any specific rules or sanctions. Most European countries implemented national legislation in compliance to the ePrivacy directive, advising website owners to inform users about the usage of cookies.Even with the EU's General Data Protection Regulation (GDPR) to notice users about the general usage of cookies, tracking did not drop significantly as research shows [39, 40, 41]. Probably, because most users ignore the notice and just accept the terms in oder to get access to the website. Another explanation is, because not all tracking cookies can be deactivated throughout the complex settings provided to the user.

2.7.7 Tools to Block Trackers

With the rise of tracking awareness of users, the amount of tracking protection tools and browser add-ons to block ads highly increased [64, 224]. In 2002, the first generation of tools such as *Adblock* disabled the display of ads. Soon after, blocking the requests of advertising (tracking) elements was implemented. The main idea of most ad blocking tools is to block the connection to known tracking servers by using filter lists of regex-based rules on tracker domain names and rules that refer to tracking elements on web pages [224]. No (known) ads or other tracking elements such as ads in *Youtube* videos, *Facebook* newsfeed ads or pop-ups will be downloaded or displayed on the website. Popular ad-block browser extensions are: *Adblock Plus*[23], *AdBlock*[24] or *Ghostery*[25]. It should be noted that the quality of ad-blocking tools will always depend on the coverage (amount of known/blocked tracking domains) and the update interval. However, there is widespread criticism about the usage of such tools to white-list specific trackers through a monetized system [61], or by the utilization of collected user data [69]. In this scenario a fee is payed by the advertisers to be removed from the tracking block-list as well as added to the white-list of "accepted ads" [68]. The accepted ads will then be displayed and used for tracking.

[23] `https://adblockplus.org`
[24] `https://getadblock.com`
[25] `https://ghostery.com/`

The white-list of acceptable ads has been updated on average every 1.5 days and grew from 9 filters in 2011 to over 5,900 in the Spring of 2015 [225]. The study on 5.000 websites showed that 59% websites allowed accepted ads. Therewith, using these ad-block tools, users can still be tracked.

Also various research has been done in the field of tracking detection [150, 12, 151, 11], showing the difficulty of preventing web tracking with third-party tools. Roesner et al. [13] offer a refreshingly differentiated look into third-party tracking on the web. They classified the different types of tracking and investigated the top 500 and 500 less popular domains for instances of tracking. Blocking third party cookies was found to be effective against most trackers, but it is not easy to avoid tracking by major social networks plug-ins without breaking their functionality completely. To that end, they developed the *ShareMeNot* browser extension that strips cookies from the initial requests to these resources, but allows them if they are clicked. However, this solution is limited to stop tracking of some services like *Twitter.com* or *Youtube.com*, while it does not entirely remove the presence of Facebook and Google. Jang et al. [226] used information flow analysis to find instances of history stealing and behavior tracking, i.e., precise recording of mouse movements within websites. They found that 46 sites of the Alexa global top 50,000 performed history stealing.

Better alternatives are tools such as: *Privacy Badger*[26], *Disconnect*, *uBlock*[27] and *uMatrix*[28]. While *uBlock* and *Disconnect* block trackers mostly by using black-lists, *uMatrix* uses a more advanced method to block all (third-party) requests of a website. Those tools cover not only basic tracking elements like cookies, images or scripts, but also XMLHttpRequest (XHR), Frames, plug-ins and other elements.

Privacy Badger follows the idea of a learning algorithm to identify and block tracking elements through their appearance while browsing the web. Here, all elements that appear on more than one websites are categorized as trackers and potentially blocked. In addition to that, the user can manually adjust the settings and block those tracking elements that have not yet been marked as trackers but are present on the website. However, the effectiveness of blocking tools is limited due to the lack of usability. If the tool blocks content that a website needs to display a specific feature, like using Google API[29] for visual effects, the website will not load correctly. In this case, the user needs to adjust the settings or temporarily disable the blocker, allowing tracking.

In 2016, advertisers estimated a monthly revenue loss between $3.9M and $120K [64] on ten large publisher sites. With the loss of revenue trough ad-blockers not displaying advertising, many websites implement anti ad-blocker scripts forcing the user to disable their ad-blocker [65, 66, 67]. Here, the website tries to detect the absence of known ads and scripts while ”bait” ads check whether a fake ad gets blocked. Alternatively, side channel timing effects are being checked that might occur due to the ad blocking code.

As a consequence, ad-blockers make increasing use of filters to detect such scripts and circumvent them. In the last years, the number of filter lists has grown to nearly 2000 [66]. But as resent research shows [62, 227], however, even advanced tools can not protect against all trackers and exposed the user to 10-30% of trackers.

Other research suggests, on the other hand, to detect and filter tracking by analyzing ads on their visual features [228]. But the solution still needs more profound work to solve basic image obfuscation.

2.7.8 Anonymity Networks / IP Hiding

Users seeking for tools to hide their IP address to provide anonymity make use of tools such as Virtual Private Networks (VPNs), anonymous proxy servers, or Tor[30] and JanDo[31]. Some proxy services can also

[26] https://www.eff.org/de/node/73969
[27] https://github.com/gorhill/uBlock
[28] https://github.com/gorhill/uMatrix
[29] https://developers.google.com/apis-explorer
[30] https://www.torproject.org
[31] https://anonymous-proxy-servers.net/en/jondo.html

remove cookies and other tracking elements from HTTP requests or are used to bypass censorship [229]. As we describe next, most tracking techniques can not be stopped using a VPN. Furthermore, one needs to make sure that the proxy service does not save the HTTP requests, since this can directly be used for behavior tracking.

The Tor Project aims to provide anonymity against network-based surveillance by encrypting and routing all TCP-based traffic through multiple Tor nodes. It can provide confidentiality against attackers on the user's network, because the data is sent encrypted to the *entry node*, which re-encrypts the data and routes it further through the Tor network. Tor also offers "so-called" hidden services, i.e., web services reachable only via the Tor network providing anonymity and confidentiality for both, client and server (details how Tor operates can be found online [230]). At this point in time, the Tor browser is the only browser enabling additional fingerprinting tracking protection features by hiding or disabling specific browser features and deleting cookies after each session. Unfortunately it has its limitations and implementation flaws, which are discussed in Section 6.1.

A problem with Tor is, that there is no protection against malicious *exit nodes* as the servers should receive the packets just as if the client had sent them directly with a different source address. Threats are global attackers and leaks by local software. Global attackers can use simple statistics or advanced techniques [231] to de-anonymize users. Local software, for example by using the File-Transfer-Protocol in active mode, sends the local IP address as part of the protocol[32]. Some BitTorrent clients use this method, together with UDP[33], which is not supported by Tor.

2.7.9 Anonymous Search Engines

As mentioned in Section 2.5.2, search engines can track user behavior. With *DuckDuckGo*[34], *Startpage Search Engine*[35], and *Ixquick Search Engine*[36] a variety of alternative browsers exist claiming not to collect any private data. In addition, those search engines hide the HTTP referrer header which disallows websites receiving the used search string.

2.7.10 User Driven (Academic) Advertising Models

Even though users are concerned about their privacy, studies show that about 50% would still be willing to share personal data with companies in exchange of money [232, 233, 234]. Motivated by these findings, researchers developed various systems [235] to restructure the distribution of ads. The idea was to ensure more privacy by stopping tracking, but also keeping ecosystem in which publishers get paid either by the consumer directly or by advertisers for displaying relevant ads.

The first architecture to provide privacy preserving advertising was proposed by Juels [236] in 2001, suggesting a client side proxy to manage user profiles and private information retrieval as well as distribution of ads. The usage of mix networks enabled anonymization. The design provided the basis for a broad range of further research which is detailed in the following. The system Juels proposed was not designed to retrieving ads in real time, making it impractical in todays web. Aiming to solve the shortcoming of Jules, Riederer et al. [237] proposed a browser plug-in and proxy based network allowing the user to decide what personal data is shared in order to receiving compensation for it.

With Adnostic, Toubiana et al. [238] proposed an architecture enabling ad targeting without compromising the user privacy to the ad-network. By profiling the user on visited websites, the system decides within the browser which specific ads, out of a set send by the ad-network, are most relevant to the user's

[32] https://trac.torproject.org/projects/tor/wiki/doc/TorifyHOWTO/FTP
[33] https://blog.torproject.org/blog/bittorrent-over-tor-isnt-good-idea
[34] https://duckduckgo.com
[35] https://startpage.com
[36] https://ixquick.com

interest and be displayed. The profiling is done by categorizing the visited websites on their meta-data (keywords, description, title) and URL. Accounting in the "charge per click" model remains unchanged since it is not a privacy violation for the ad-network to know what ad a user clicked. For the "charge per impression" model, the correct advertiser must be billed without the ad-network learning which ad was displayed to the user. Here, Adnostic suggests an additively homomorphic encryption and efficient zero knowledge proofs. Limitations are on the higher network usage, possible attacks as well as the low quality of user profiling since the information gathered by Adnostic might reflect less personal information as the ad-networks could gather.

Privad [239] is a similar concept using a sandboxed agent running locally to profile all user activities and serving most relevant ads sent by an intendant proxy. The proxy is responsible for billing and ad distribution which is done by encrypted and anonymized connections between the client and ad-broker, securing the client against tracking as well as attacks. The proxy system is prefetching a set of ads to be served to the user. The agent will then select most relevant ads based on the local profile. Problems arise when attackers de-anonimize the user information [35, 37, 36, 14, 6] or the dealer is colluding with the ad-brocker. In this scenario the user has no control over the shared data.

A different approach is presents by RePriv [240], a system for controlling the release of private information within the browser. In contrast to Adnostic and Privad, the system enables the user to decide which parties may access the various types of data stored inside the browser. Third parties can request private information through a policy based code, collecting various website information like the topic or DOM elements. Limiting factors are the overhead for users to get assess to every new services requesting profile data, higher network timing and the development of data mining policies for ad-companies to gather the profile data.

ObliviAd [241] presents a novel architecture protecting the privacy of the user by implementing a secure hardware-based private information retrieval (PIR) protocol for distributing advertisements. For billing, high-latency mixing of electronic tokens are used without disclosing any information about client profiles to brokers. The proposed protocol fetches an ad, then the user sends his profile in an encrypted form to a secure coprocessor that resides on the broker side. Since ObliviAd does not assume any anonymous channel, the broker may learn the identity of the user, but not the user profile. The secure coprocessor selects and sends the advertisement in an encrypted form that best fits the user profile according to the algorithm specified by the broker, but preventing the broker from learning any information about the selected advertisement. By the strong security protocol, ObliviAd's drawback is its increase in complexity and deployment. Further, as most of the suggested systems, this approach does not guarantee protection against fingerprinting.

With the work done in MyTrackingChoices [242], and MyAdChoices [243], researchers propose systems in which the user can decide which website topics/categories (e.g., health, religion, politics or finance) are privacy-sensitive to them and should be blocked from tracking. The tools allow users to specify the categories of websites that are privacy-sensitive and block the trackers present on such websites only. Although the tools elaborate control over ads, they do not prevent a consequent as well as extensive tracking protection, making tracking still possible to profile the user.

2.7.11 Tracking-Free (Paid) Services

Internet users need to understand that publishers and content creators require compensation for their services to cover the cost associated with its provision and delivery. Costs, that most people do not see or are willing to pay when they do not see their benefits [244]. But in todays web the old saying "if you're not paying, you are the product" holds true in case of publishers using tracking ads to sell user data. In the following, various tracking-free Internet usage-models will be outlined.

Subscription Based Plans:

Subscriptions are offered by many services, apps and websites to exclude advertising against payment. But users do not like to spend time on subscriptions to every new service/website or spend money on a one time service like reading a news article [244]. This is why it is difficult to get a constant revenue with this method, which is not practical for most users as well as services.

Micropayment Systems:

Micropayment services like Flattr[37] provide an alternative to subscription-based systems. Flattr was launched in 2010 and bought by *AdBlock Plus* owner Eyeo in 2017. Flattr pays content creators automatically the amount depending on the attention the user is giving to specific content. This is done using a browser add-on calculating the engagement time and mouse action on a URL (or video/podcast). The algorithm calculates the payable amount while only transmitting the URL to the service provider. This keeps all the private data used for the decision on the local device. With a minimum fee of $3 a month, 90% of the amount gets split up with the content creators that the user is supporting (visiting). Similarly, with the Google Contributor[38] a big advertising player is researching on this concept to reduce the amount of ads delivered by advertising services. Users registered with this service would have pay a monthly fee to support the ad free sites. However, so far, only few publishers participate.

Within Browser Models:

The first model was proposed by Mozilla in 2014 when Subscribe2Web[39] was introduced to eliminate the web's dependency on ad-based financing. The aim was to provide a general web API accessible from any browser through which users would pay a monthly subscription fee in exchange for accessing ad-free content. Unfortunately this project was discontinued some time later.

The next was the Brave[40] open-source browser (based on Chromium), launched in 2016 with the built-in feature to strip out all ads and trackers. In 2018, Brave announced an ad replacement and revenue sharing program with its pay-to-surf business model. Users opting in will then receive ads sold by Brave in replacement of the ads blocked by the browser. Based on the attention the user spend on sites the browser automatically calculates the reward, which will be received in BATs (Basic Attention Token). BATs are a utility token in a new, blockchain-based digital advertising and services platform building on the Ethereum[41] technology. It is used as a unit of account between advertisers, publishers, as well as users on the BAT platform and can be utilized to directly measure, exchange, and verify attention [245]. The revenue on advertising is shared between publishers (55%), ad partners and users (15% each). While user targeting is done inside the browser by analyzing the anonymized browsing history, billing is done using the *Zero Knowledge Proof* technology Anonize [246]. The concept of Brave is interesting, but placing own ads in front of the original is being discussed controversially [247].

Ad-Free Browser:

An alternative to Brave is the Chromium-based browser Epic[42], launched in 2013. The features list of the privacy friendly browser includes: removing all ads and tracking elements, not passing the referrer header data, using an Chrome user agent string, all search requests through a build-in VPN-proxy service, URL auto-suggest and language translation are deactivated on default, automatic selection of SSL connection and active Do-Not-Track-Header. A major drawback is, that because of blocked third party elements, various websites will not work with Epic.

[37] `https://flattr.com`

[38] `https://contributor.google.com/v/beta`

[39] `https://drive.google.com/file/d/0BzwYn5IpP3wtbmhPMEdNV1Ayb1k/view`

[40] `https://brave.com/`

[41] `https://www.ethereum.org/`: Decentralized platform that runs smart contracts (apps) on a custom built blockchain - a shared global infrastructure that can move value around and represent the ownership of property.

[42] `https://www.epicbrowser.com`

Network Level Ad-Blocking:

In 2015 Internet service providers (ISPs) started to block advertising on the network level. The U.K. mobile carriers *EE*, *Three*, *O2* and the Caribbean mobile operator *Digicel* announced to work with the start-up company *Shine* to implement network-based ad-blocking features [248]. However, a few months after the EU net neutrality rules making ISP ad-blocking illegal across the EU end of 2016, Shine switched their business model into an advertising verification service.

At the moment the open source software Pi-hole[43] is the only real alternative. It is designed to run on embedded devices with network capability such as the Raspberry Pi[44]. The linux-based network-level advertising and tracker blocker acts as a DNS sinkhole for blacklisted domains on a private network. Ads and trackers are blocked before entering the network. This enables blocking ads also on mobile phones, smart TVs, third-party apps, and streaming video services. User reports state a 30-60% reduction of network traffic [249, 250]. So far, the ad industry has not taken any actions against Pi-hole since its launch in 2015, most likely due to the low number of users.

2.8 Summary

On the enormous amount of web tracking techniques, various privacy protection mechanisms try to reduce tracking. However, as we pointed out in this chapter, all of the tracking protection mechanisms have their weaknesses and are not able to prevent the leakage of privacy sensitive data on the web. With the risks, threats and implications of web tracking we presented in this chapter, like price discrimination and de-anonymization of users, there is a need to provide better solutions to reduce tracking and prevent the leakage of privacy sensitive user data on the web. Motivated by open challenges in various protection techniques, we developed and evaluated new concepts to stop web tracking, presented in the next chapters.

[43] `https://pi-hole.net/`

[44] `raspberrypi.org` : A small single-board computer.

3 Automated System to Detect, Analyze and Protect Against Tracking

In order to understand the tracking ecosystem and to provide tracking protection, we built an automated system to detect and evaluate tracking elements. In this chapter, we present the system architecture, mode of operation and experimental setup. We give insights in the usage of different tracking techniques and evaluate their appearance on websites as well as the privacy implications for users. The results are shown in Section 3.5.

As already discussed, web tracking protection depends on blocking a broad range of tracking methods, the known tracking domains, and providing a mechanism to keep this list of blocked domains up-to-date. Facing these challenges we implemented a system to scan websites for different tracking techniques and extracted the tracking domains. Moreover, our systems export a public available *Tracking-Protection-List (TPL)*[1] that can be used as an ad-block-list[2] for the Internet Explorer (Version 9-10).

Because tracking is mostly done in the background, without the user's notice, it is crucial to us making our findings understandable for users. To cover this, we visualized and explained our results on a website publicly available at `https://www.sit.fraunhofer.de/de/track-your-tracker/` as well as in our *Web Tracking Report* [22].

3.1 Architecture and Implementation

Our tracking detection system, shown in Figure 3.1, is based on a crawler using the Internet Explorer as browser to open websites and an analysis algorithm to detect as well as extract the tracking domains. We choose the Internet Explorer to analyze the tracking elements that are specifically rendered in the browser's HTML DOM and therefore guaranteed to be blocked by the TPL. The control program of the crawler is implemented in Java and makes use of a Squid Proxy[3], together with an ICAP server[4] to capture and filter the HTTP traffic. The data is then stored in a MySQL[5] database. When compiled to a *JAR* file, the system is running on a standard Windows computer connected to the Internet. To analyze the HTTP traffic the system uses the pre-installed Internet Explorer (Version 9-10) to open a predefined list of websites in an automated manner.

The list of website URLs is extracted in a previous (separate) process using an URL crawler and is updated every month. This URL list unifies the most frequently visited websites ranked by *Alexa* (Top 500 websites in Germany[6]), *Netcraft* (Top 100 websites in Germany[7]) and the list of websites obtained from the *Information Community for the Assessment of the Circulation of Media (IVW)*[8]. In order to reduce crawling time, we only selected the top ranked German websites.

On each website the system randomly opens additional ten internal links[9] in a new window to scan for additional tracking elements. While the HTTP traffic is routed through the Squid Proxy and forwarded to the ICAP server, the tracking detection algorithm analyzes the traffic. The tracking evaluation is done on the fly and is discussed in detail within Section 3.2.

After analyzing the predefined list of websites, the gathered tracking domains are extracted from the database into a format that is readable by Internet Explorers integrated feature "Tracking Protection List"

1 `https://blogs.technet.microsoft.com/iede/2011/04/04/tracking-protection-lists-und-ihre-updates/`
2 `https://www.microsoft.com/de-de/iegallery`
3 `http://www.squid-cache.org`
4 `https://tools.ietf.org/html/rfc3507`
5 `https://www.mysql.de`
6 `http://www.alexa.com/topsites/countries/DE`
7 `http://toolbar.netcraft.com/stats/topsites?c=DE`
8 `http://ausweisung.ivw-online.de/online`
9 We select only links pointing to the website visited at the moment, in order to detect further tracker embedded within this website.

(TPL)[10]. Then, the TPL file is uploaded to a public server, where users can download and use the list to block the listed tracking domains[11].

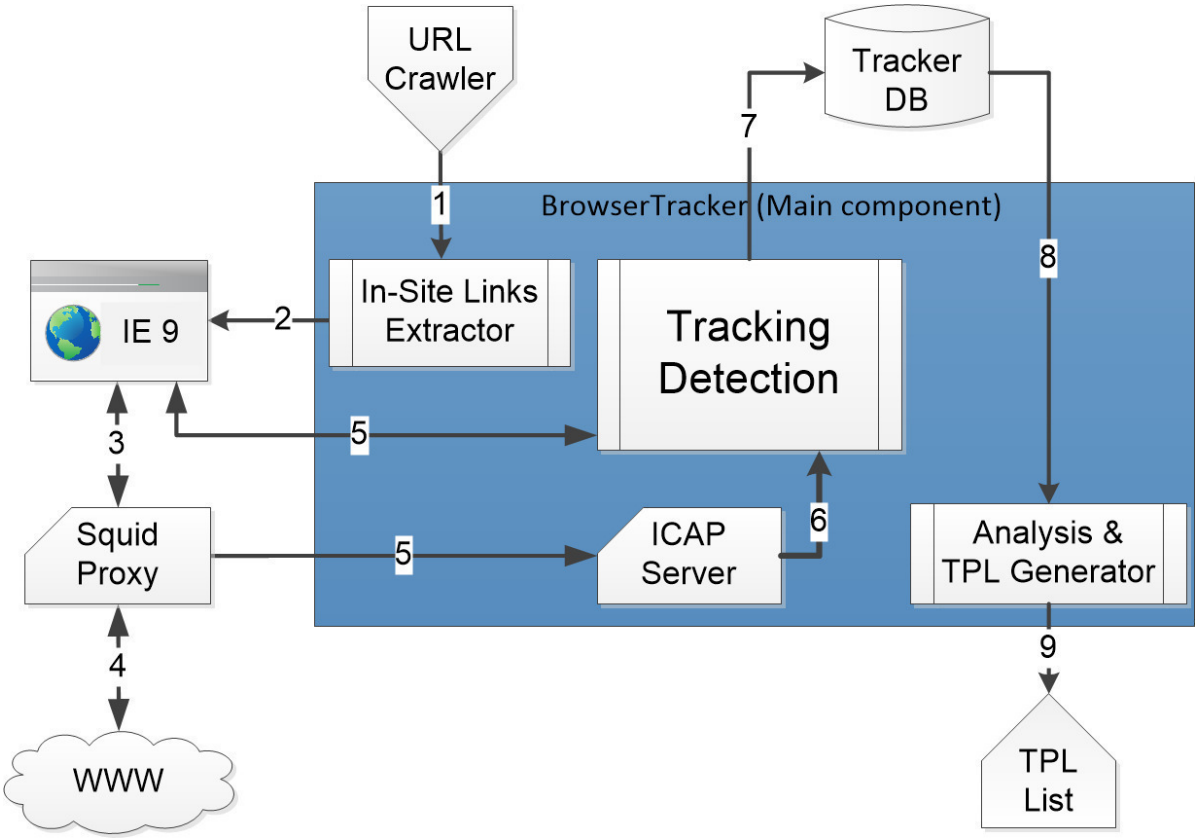

Figure 3.1: Tracking detection and protection system architecture

Mode of Operation:

The mode of operation is depicted in Figure 3.1. The main component (blue), exported as *BrowserTracker.jar*, contains the web crawler scanning the websites, the HTTP traffic processing module, the tracking detection algorithm and the TPL file generator. In the first step (1), the *In-Site Links Extractor* module takes one website at a time from the URL list and opens it in the Internet Explorer (step 2). Then, by scanning the HTML code of the website, this module generates a random list of up to ten additional internal links that shall be opened as well as scanned next.

All HTTP requests and responses are processed by the *Squid proxy* and forwarded to the *ICAP server* for further processing (step 3-5). The *ICAP server* evaluates the HTTP traffic in real-time (steps 5-6). The *Tracking Detection* module then uses the built-in metrics to detect tracking elements by analyzing the HTML website data and the HTTP traffic (steps 5-6). The results are saved in the *MySQL database* (step 7).

The *Analysis and TPL Generator* module evaluates all tracking elements and exports the tracking domains into a TPL readable text format (steps 8-9). The TPL file is then uploaded to a public server.

The following steps describes the process in more detail (function names are printed bold):

- **URL crawler** generates a list of domains to be scanned by the crawler
- **BrowserTracker** processes the web traffic data (main module)

[10] https://blogs.technet.microsoft.com/iede/2011/02/09/ie9-tracking-protection/
[11] https://www.sit.fraunhofer.de/tpl/

The main component (BrowserTracker) is responsible to start the **IcapServer**, connect to the proxy host and setup the **Squid server configuration**. It then opens a database connection using **TPLDBCon**. Next, the crawling process is started by opening one website after the other to scan for tracking elements. Here, the browser now sends all requests to the proxy server, which are relayed to the *ICAP server*. Now, various **TrackingDetection** modules evaluate the traffic data and HTML source code for tracking elements. All detected trackers are then inserted into a result list and saved using **TPLDbCon**. The last crawling step is to click on links, in order to open additional pages until the page limit is reached.

- **TplStatistikCode** generates a TPL file by reading the data from the DB

3.2 Tracker Classification and Detection Metric

Through the evaluation of the most commonly used tracking techniques in a two month long pre-study, we implemented the detection of the following tracking methods:

- HTTP cookies (setCookie header, JavaScript)
- Flash LSO
- HTML5 Local Storage
- Ad images and beacons
- Third-party scripts
- Third-party iFrames

We classified potential tracking elements as tracking if:

1. Cookie is stored as third-party
2. All other elements:
   a) The URL of the element appeared on at least two internal web pages AND
   b) The URL of the element was present on at least two other websites (of all scanned domains)

Rule 2.a) and b) exclude elements that are only used on the actual website (internal website elements).

3.3 Long Term Evaluation and Results of Web Tracking on German Websites

We evaluated the HTTP traffic data of 1,634 websites during November 2012 and January 2014. The goal was to identify and analyze different third-party tracking elements as well as their occurrence in order to develop efficient protection mechanisms.

The research questions for the evaluation were:

- How many trackers are embedded on websites?
- What are the top 25 tracking domains?
- How many websites do trackers cover (amount of websites a tracker can re identify the users browsing on the web)?
- Are tracking methods changing over time?

- What is the usage of different tracking methods?

In this chapter we present our findings on studying the appearance of tracking elements embedded on websites. First, we summarize significant results. Subsequently, we provide statistics on the appearance of trackers: We show their capabilities to establish detailed user browsing profiles by the amount of websites the trackers are embedded on. We give insights on the total amount of websites the trackers are able to monitor during the entire data collection and a single browsing session. We then turn the view on the embedding websites, and show how many different tracking elements the websites embed on average, during the entire crawl or a single browsing session.

3.4 Experimental Setup

During the time of the evaluation (November 2012 - January 2014), the crawling process was manually started several times a month. In each run, the crawler visited the generated list of websites and additional 10 random internal pages of each domain. Because of the amount of websites and the network speed, each run took two to three days to complete. The crawler was executed in the network of TU Darmstadt with no restriction to the Internet. To prevent dead links, the crawler generated a new list of websites to be scanned every month.

3.5 Results

In the entire evaluation period we identified 1,209 distinct tracking domains. Among the top 25 websites with the highest amount of trackers, on average 45 tracking elements were embedded per website.

The tracker with the highest capabilities to establish a browsing profile was *doubleclick.com*. In a single crawl, *doubleclick.com* was able to monitor 971 websites. After the evaluation time, we found out that the tracker was embedded on 1,463 websites, covering 90% of the scanned websites.

3.5.1 Top Tracker of Entire Evaluation Period

During the 16 months of our experiment we summed up the amount of websites the trackers were able to monitor the users browsing habits. Figure 3.2 visualizes the top 25 trackers with the highest amount of websites they are embedded in and therefore can directly monitor the users browsing profile.

Ranked highest, *doubleclick.net* is embedded on 1,463 websites, covering 90% of the scanned websites (aggregated websites we visited in entire crawling period) and having the most capabilities to establish a detailed user browsing profile. The top four trackers are embedded in nearly 70% of websites, and on average, the top 25 trackers are able to track users on each scanned website.

3.5.2 Top Tracker in a Single Crawl

To show the capabilities of a tracker in monitoring the browser activities during a single browsing session, we use the data of a single crawl performed in January 2014. Visualized in Figure 3.3, the top trackers in this crawl are *ivwbox.de* and *doubleclick.net*, followed by the tracking companies *adition.com* and *nuggad.net*. While *ivwbox.de* was embedded on 994 different websites, the average amount of websites in the top 25 trackers are able to monitor 330 websites during a single browsing session. In this single crawl from January 2014 we identified in total 633 different trackers. Notice, the amount of trackers varies from each crawl and depends on the additional 10 randomly selected internal pages, and the time of day the crawl was performed.

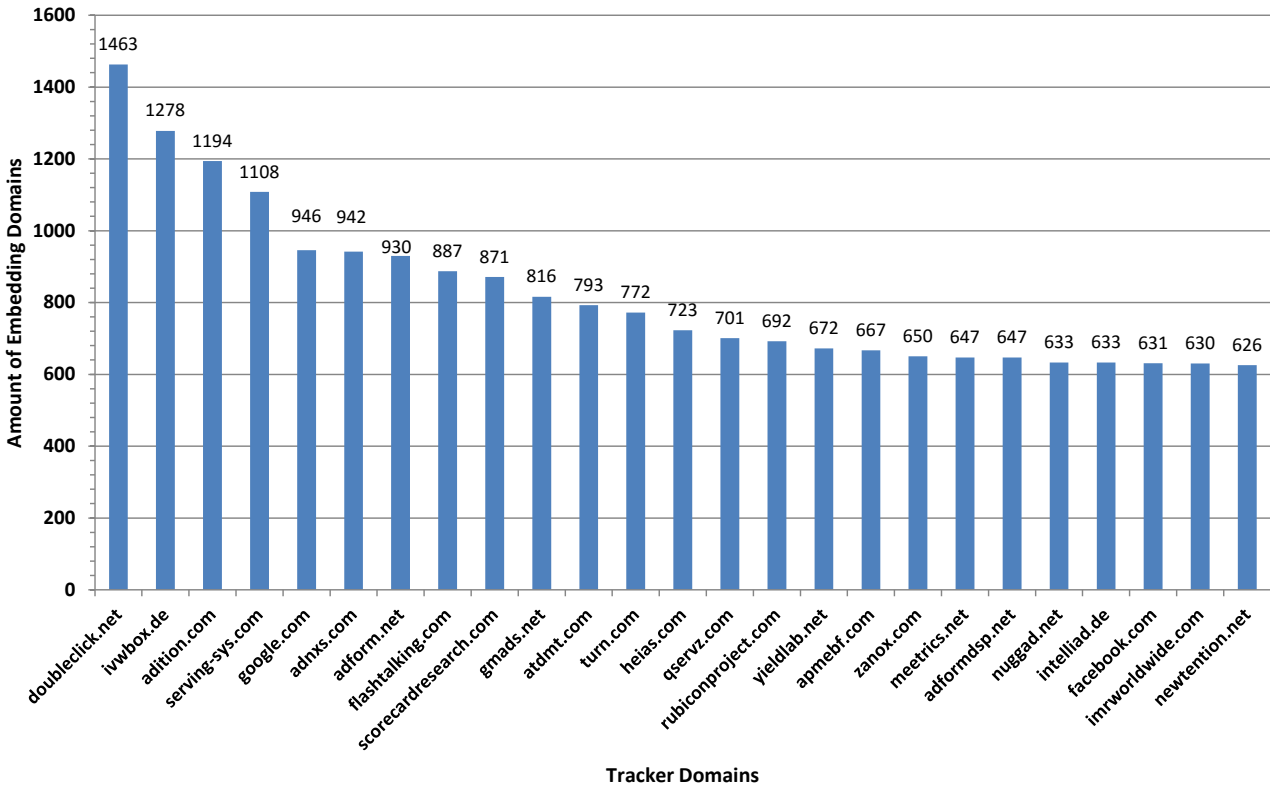

Figure 3.2: Top 25 trackers of entire evaluation period with the amount of websites they can monitor.

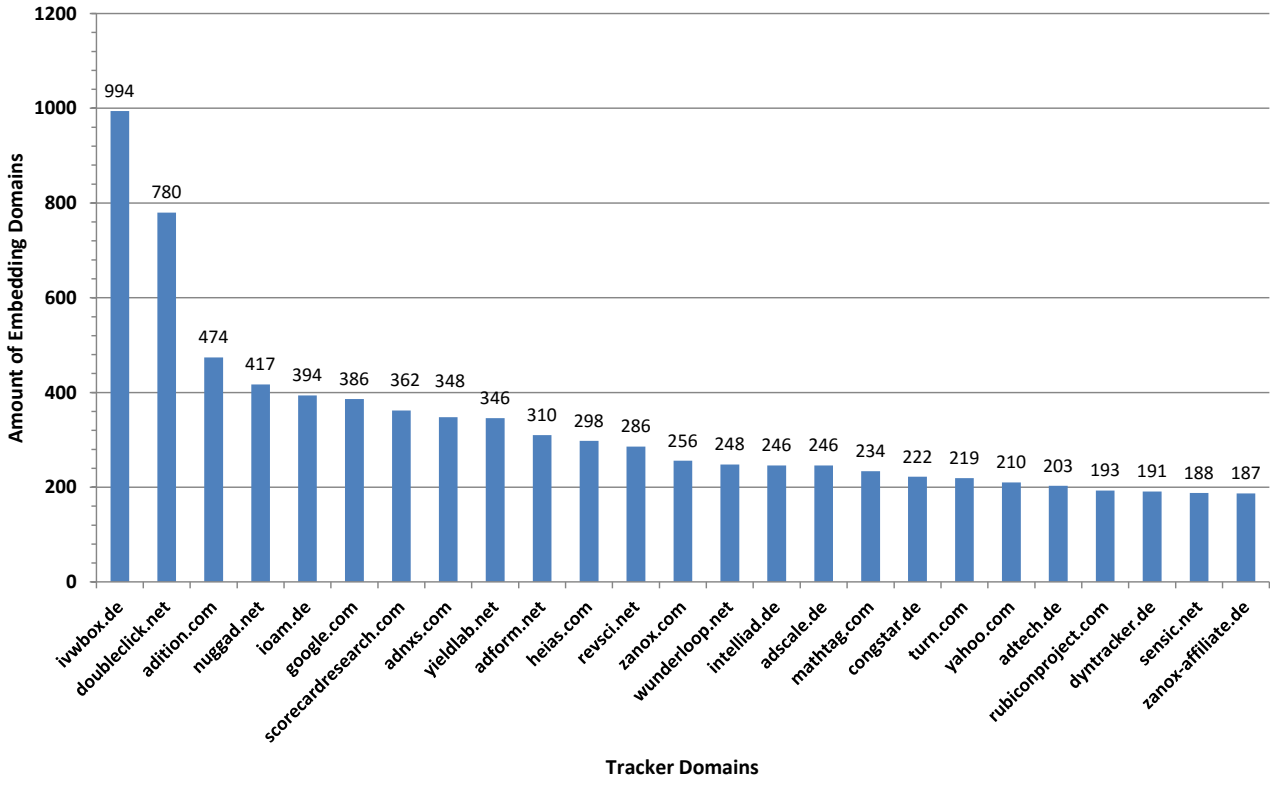

Figure 3.3: Top 25 tracker in a single crawl from Jan. 2014 with amount of websites being embedded in.

3.5.3 Highest Tracker Appearance in Different Crawls

The tracking potential of a single crawl, and therefore a single browsing session, depends on how many websites are being visited and the time of day when visiting. To compensate the results of a single crawl we consolidated different crawls to show the highest amount of websites a tracker was embedded in during a crawl.

Figure 3.4 shows the highest appearance of trackers taken from different single crawls. The top ranked tracker *ivwbox.de* was embedded on 1,110 websites in the crawl of May 2013. Ranked second and third are the trackers *doubleclick.net* with 971 websites and *adition.com* with 741 websites being embedded on. The high amount of websites that *ivwbox.de* was embedded in can be explained by the high amount of websites taken from the *ivw-online.de* list that have been merged to the crawling list of websites.

In the set of all 1,634 crawled websites, the average amount of trackers embedded in websites was 13.74.

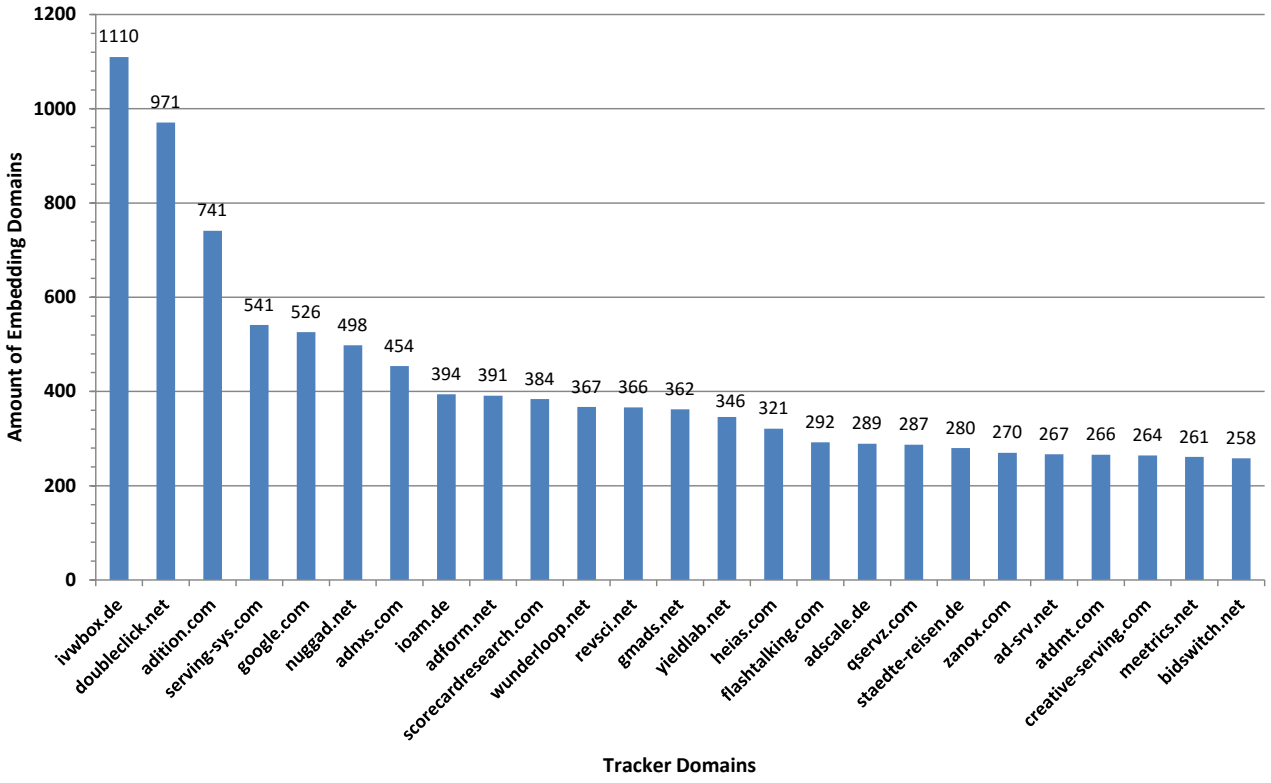

Figure 3.4: Highest amount of embedded trackers in crawls between November 2012 and January 2014.

3.5.4 Total Amount of Tracking Elements on Websites

Figure 3.5 shows the top 25 websites with the total amount of different tracking elements that are embedded on their website.

On average the top 25 websites embed 153 different trackers. An interesting fact is that overall more then half of the top 25 websites belong to online-news media websites. These findings suggest that news websites tend to embed far more trackers then other websites.

We conclude that many websites use various tracking techniques from different tracking domains. Alternatively, this indicates that the tracking companies regularly change their tracking domains, and

therefore appear as a new tracker in our dataset. Since we could not find any evidence that would allow us to link tracking domains to tracking companies we cannot provide more insight here.

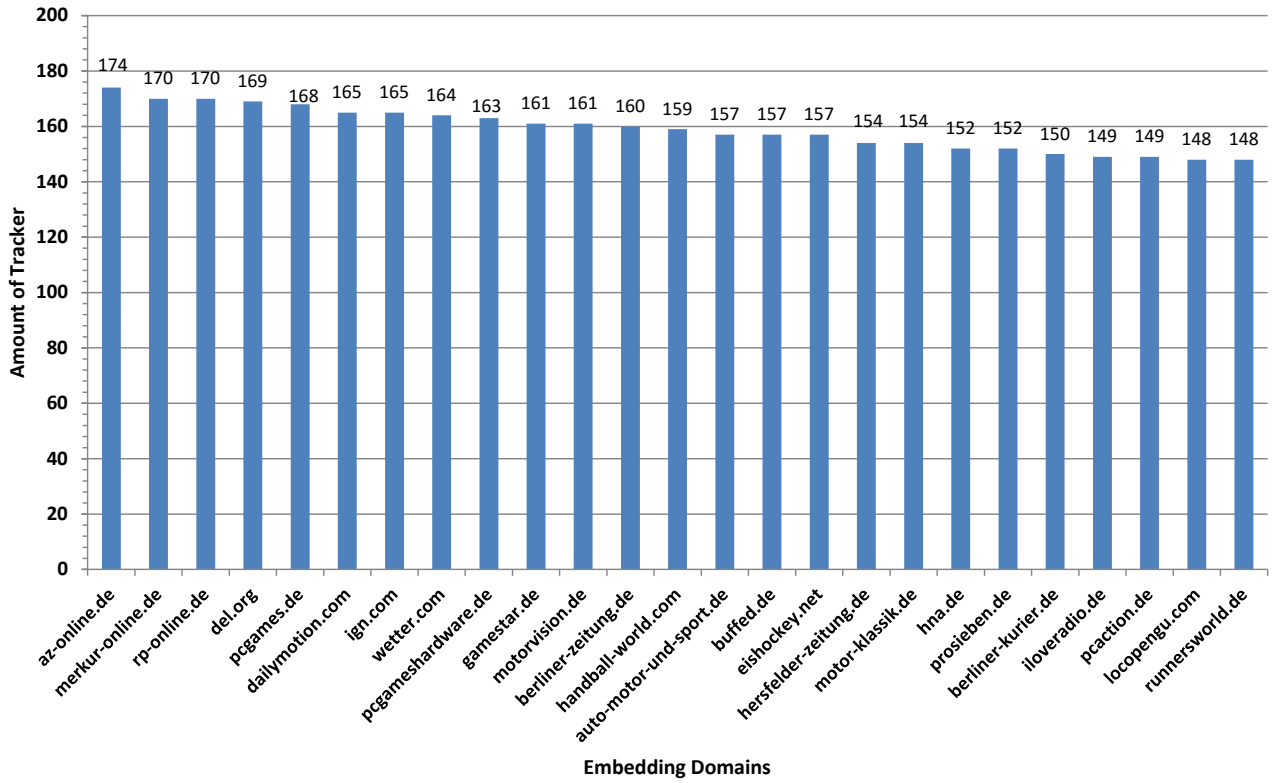

Figure 3.5: Top 25 websites with different tracking elements observed during the entire crawling period.

3.5.5 Amount of Tracking Elements on Websites in a Single Crawl

To show the appearance of tracking elements a user would experience while visiting various websites during a browsing session, we use the results conducted during a single crawl. Figure 3.6 presents the top 25 websites with the amount of tracking elements that were embedded in a crawl done in January 2014. In this crawl *rp-online.de* embedded the highest amount of tracking elements. Note, that according to our tracking detection algorithm, the results derive from scanning the website with additional 10 internal pages. The amount of trackers can vary depending on the visited internal pages. For the crawl of January 2014, on average over 50 tracking elements were found in the top 25 websites.

3.5.6 Average Amount of Tracking Elements Embedded on Websites

Figure 3.7 shows the average amount of trackers we found on websites during the entire crawling period. The top 25 websites embed on average 45 trackers. The average amount of trackers embedded on websites in a set of 1,643 websites was 8.68.

3.5.7 Maximum Trackers on Websites in Different Crawls

Similar to Section 3.5.3, we provide the evaluation of the highest amount of trackers found on websites during the entire crawling period. For each website we extracted the crawl with the most embedded tracking elements. The list of top 25 websites embedding trackers is presented in Figure 3.8.

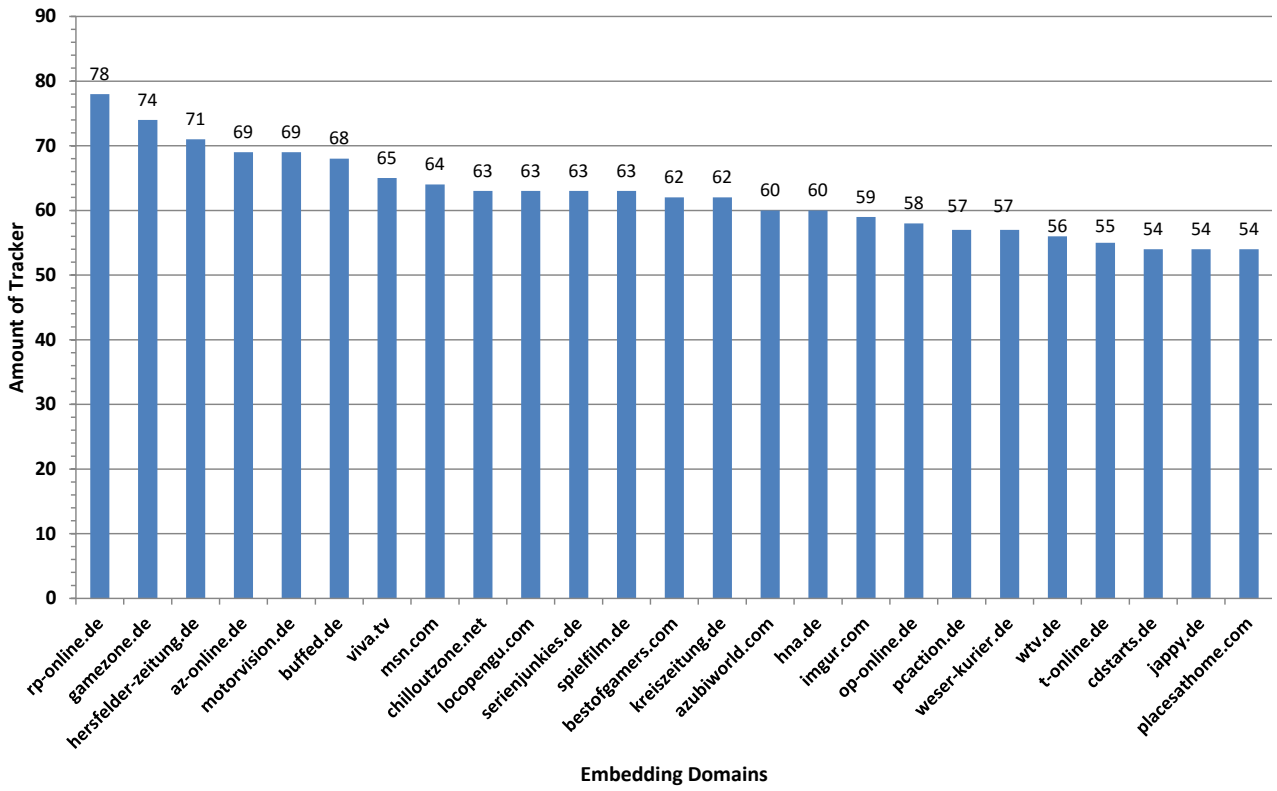

Figure 3.6: Top 25 websites embedding tracking elements in a single crawl in January 2014.

Again, we notice that the amount of trackers being embedded on websites changes during the crawls, which could be explained by the time of day we accessed a website. On average we found 84 tracker for the top 25 websites.

3.5.8 Evaluation of Tracking Methods

Figure 3.9 presents the usage of different tracking methods that we identified during the entire browsing period. It shows that cookies are used most for tracking, followed by third-party scripts, ad-images and web beacons (tracking pixels). Note that the support for detecting Flash and HTML5 tracking was added in January 2013. The increase of tracking elements in May 2013 is explained due to the fact that we added more websites to the list of scanned domains in order to cover a wider range of most frequently visited websites. We also can not provide statistics for the usage of HTML5 tracking elements for December 2013 and January 2014 because of an Internet Explorer update that changed the internal handling of HTML5 elements. Besides that, we notice a constant appearance of tracking elements during the entire time.

3.6 Summary

In this chapter we presented insights of various tracking techniques used on the web. Visualizing the huge amount of tracking elements on websites, showing that the top 25 websites embed 153 different trackers, implicates a privacy problem for users being tracked while browsing online. Google for example can track the user on 90% of the crawled websites and being able to establish a detailed browsing profile with the users' interests. In the next chapter we extend the tracking analysis on further tracking techniques and present an up-to-date evaluation. In addition, we show the interconnection of trackers within a network graph, cluster redirect trackers and demonstrate their possibility to share user data with other trackers.

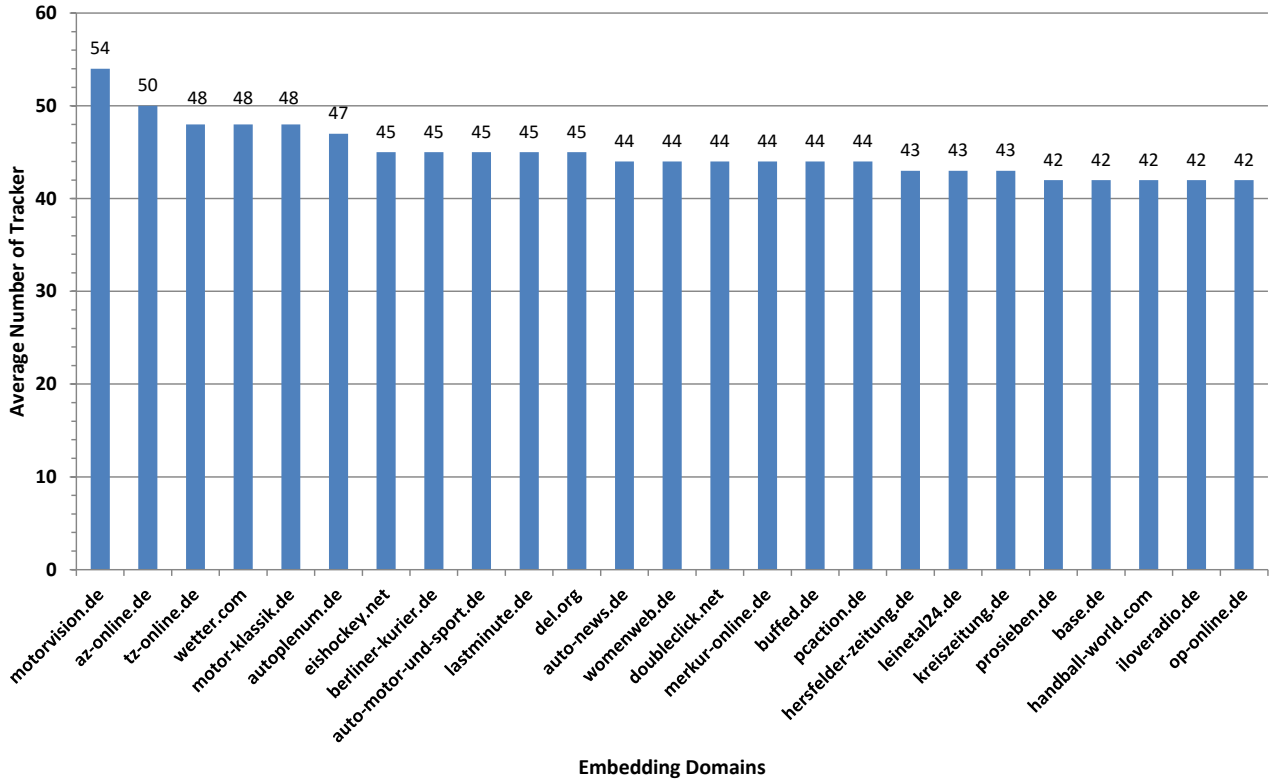

Figure 3.7: Top 25 websites with average amount of embedded trackers during the entire crawling period.

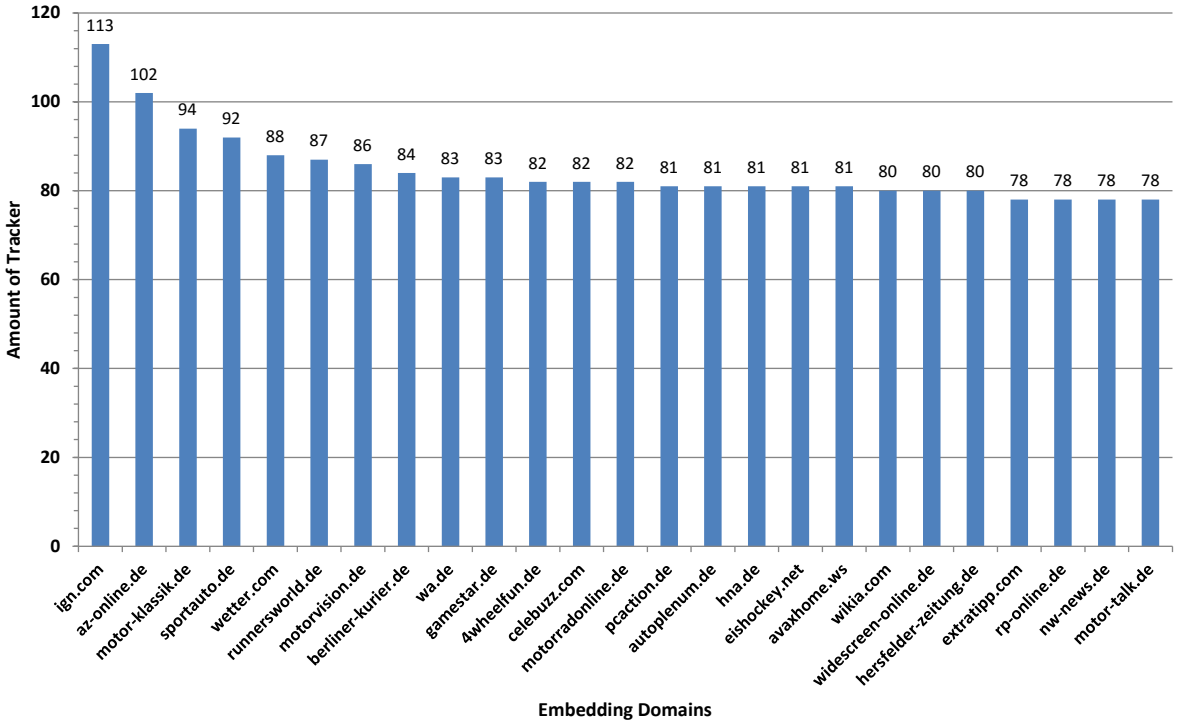

Figure 3.8: Top 25 websites embedding the highest amount of trackers in single crawls out of entire observation period.

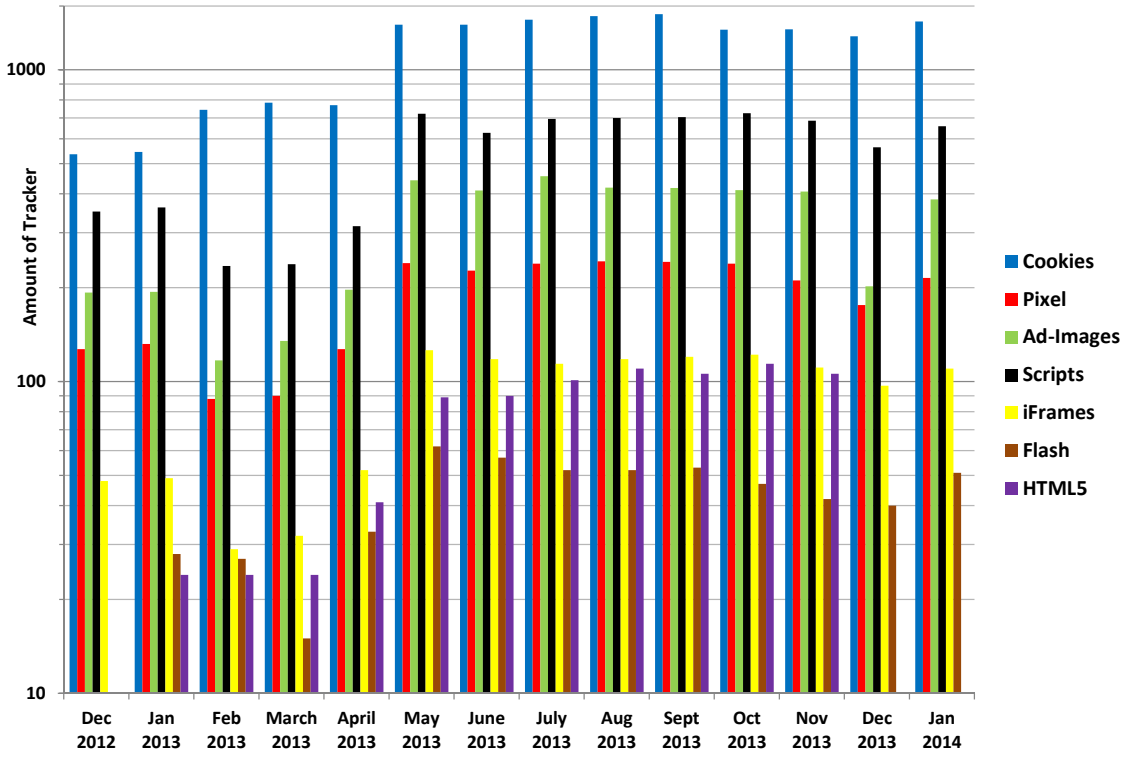

Figure 3.9: Usage of different tracking methods.

4 Dynamic Redirect Link Tracking Evaluation

Preventing the information gathering on users' online behavior, various privacy tools are available that try to block third-party elements. However, there exist various tracking techniques that are not covered by those tools, such as redirect link tracking, introduced in Section 2.6.5. Here, tracking is hidden in ordinary website links pointing to further content. By clicking those links, or by automatic URL redirects, the user is being redirected through a chain of potential tracking servers not visible to the user. In this scenario, the tracker collects valuable data about the content, topic, or user interests of the website. Additionally, the tracker sets not only third-party but also first-party tracking cookies which are far more difficult to block by browser settings and ad-block tools. Since the user is forced to follow the redirect, tracking is inevitable and a chain of (redirect) tracking servers gain more insights in the users' behavior.

By studying HTTP traces on months long crawls of 50k international websites we noticed that many trackers use HTTP redirect techniques hidden in website links, to detour users "intended" connection (link destination) through a chain of (third-party tracking) servers, before loading the intended (legitimate) link destination. Because the third-party server is opened directly, it enables advertisers to store on each redirected page first-party tracking elements on the users' machine without notice, circumventing third-party blocking tools. In addition, by using redirect links the trackers can collect valuable meta data such as the site of origin, the desired destination target, the topic/content the user is viewing and other information. Moreover, since the redirect takes only a few milliseconds, trackers can redirect the user through a chain of various servers, allowing many trackers to save cookies and share the users interests.

In this chapter we present the first in depth study of redirect link tracking focusing on user privacy, giving a comprehensive insight in the behavior, patterns and used techniques. Our data set is based on full http traces browsing the top 50,000 websites from the *Alexa.com* ranking[1] over a period of a month. We modified a version of OpenWPM [29], which uses a real Firefox browser that is instrumented using Selenium and another custom add-on, which provides a very realistic browsing experience. By crawling the Alexa top 50k websites and following up to 34 page links, we recorded traces of HTTP requests from 1.2 million individual visits of websites as well as analyzed 108,435 redirect chains originating from links clicked on those websites. We evaluate the derived redirect network for its tracking ability and demonstrate that top trackers are able to identify the user on the most visited websites. In addition, we visualize the main redirect clusters and demonstrate their strong interaction in the network of 9,862 redirect nodes. We also show that 11.6% of the scanned websites use one of the top 100 redirectors which are able to store non-blocked first-party tracking cookies on users' machines even when third-party cookies are disabled. Moreover, we present the effect of various browser cookie settings, resulting in a privacy loss even when using third-party blocking tools.

4.1 Related Work

Redirects are often miss-used to harm the user without notice. Examples are phishing attacks, where the user intends to open a legitimate website, but the attacker establishes a redirect to a malicious domain [190, 191, 192, 193]. Further, attackers use redirects to deliver malware through drive-by-download attacks [194, 195, 196] and malicious advertising [107, 108, 109]. Here, the user is redirected through a chain of malicious servers until the malware is being downloaded. Those redirecting procedures make it difficult for anti-malware or anti-virus tools to track and stop the attacker.

Researchers also discuss the problem using redirects to provide spam. This includes manipulating search engine crawlers with the goal of improving a website's ranking [251], and in the context of email, to camouflage spam websites [252].

[1] `http://s3.amazonaws.com/alexa-static/top-1m.csv.zip`

Li et al. [253] examined malicious web advertising using redirects and presented a tool to detect such malware servers. The researchers evaluate redirect chains to determine if an ad server is malicious or not. In another work, researchers investigate redirects in click-fraud links [254] displayed as an invisible overlay, which makes the user click on links they did not intend to click. In all these publications the focus is on evaluating drive-by-downloads or spam content by using redirects. In comparison, we conduct a large-scale study of redirect appearance on legitimate websites (links) evaluating the privacy impact and behavior of redirects focusing on user tracking.

Guawardana and Meek [113] look into the case of click-through rates in web advertising, especially in ad aggregation networks. Since ad aggregators derive revenue from clicks on syndicated ads, they need to link to the ad network first, which then redirects the user's click to the advertiser (intended page). This work only evaluates the behavior of clicking on ads, but does not evaluate the impact of tracking servers using redirect links on legitimate websites.

Comprehensive studies [255, 227, 25, 58, 256] comparing ad blocking tools such as Ghostery, Adblock Plus or Easylist, focusing only on the overall performance of blocking tracking elements. Other studies on various web tracking techniques like cookies, localStorage and Flash were performed earlier in [13, 57, 150]. In contrast to our work, no in-dept studies on redirect behavior on websites was conducted.

Kalavri et al. [257] inspect tracking behavior on user traces collected by a web proxy to explore the resulting tracking graph. They aim is to automatically identify trackers among the third-parties, based on structural properties in the collected graph. They show that a simple nearest neighbor approach, as well as label propagation techniques perform well with this identification method. The limitation is that no privacy implications are considered on setting tracking cookies on the user's machine like we show in our work.

A similar tracking network visualization is done by Gomer et al. [258]. Here the researchers only captured a small set of websites and focused on search query results. Our dataset includes the top 50k ranked websites with the addition of following up to 34 page links, providing a much more realistic user browsing behavior as well as a broad data collection of tracker interaction. Moreover, the graph is based on Referer connections, which does not cover all relationships between the nodes [259]

Schelter and Kunegi [260] also evaluated the tracking connectivity using a large dataset from 2012. In comparison to our work, the researchers only looked at HTML source code and did not use any HTTP traces of real browsing data like in our work. The dataset in our work is based on up-to-date browser generated HTTP traces with automated user interaction like clicking on links, moving the mouse and waiting for JavaScripts to load additional tracking elements. Plus, we use a sophisticated tracker classification, not only taking third-party appearance in the HTML source code into account but also analyze the information content in cookies. Further, we compare our results to other tracking classifications such as Easylist and OpenWPM [29] to visualize the performance to the user.

In 2018, Syverson and Traudt reported [261] on the possibility of tracking using HSTS[2], were a cached HTTP connection is redirected to a secure HTTPS connection. Here, the tracker could force the browser to check cached (visited) domains [262]. As of this, browser vendors added additional tracking prevention methods to mitigate HSTS tracking [263, 264]. Since our focus was on redirect links, we did not establish a verification setup on the applied prevention mechanisms, which might be indeed an interesting future work. Yet another study by Matte et al. on handling the GDPR cookie policy banners [265] mentions an unmeasured reference of redirects being used to provide a tracking cookie within the GDPR cookie banners. With another focus in our study, we did not notice any hints on such practices while re-evaluating our data set.

Summing up previous work, even though the idea of tracking the user within redirects is known since 2006 [266] and was discussed in a simple demo in 2011 [160] as well as in [267], to the best of our knowledge we provide the first large-scale study evaluating the usage of redirects in the wild to track users. In contrast to other work, whereas tracking is evaluated on the landing pages, we evaluate

[2] HTTP Strict Transport Security (HSTS): Security policy mechanism for websites being accessible only via a secure connection. `https://tools.ietf.org/html/rfc6797`

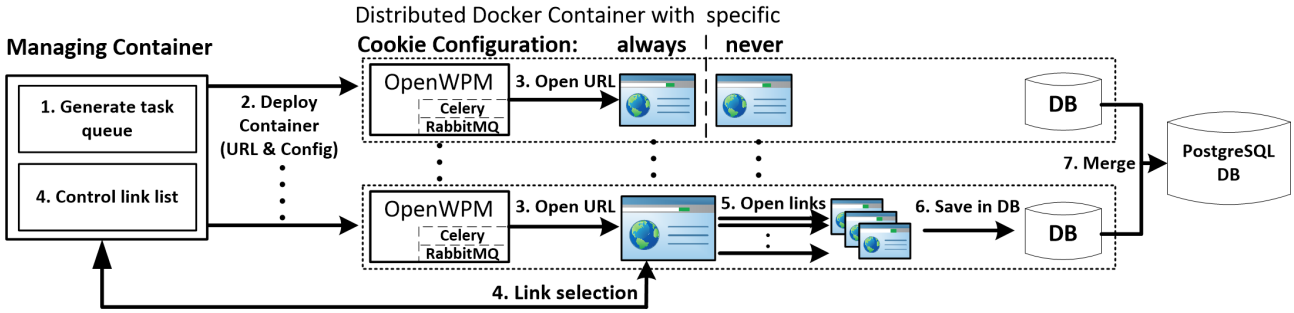

Figure 4.1: Experimental set-up.

tracking when performing user actions such as clicking on links, opening several pages, move the mouse and waiting for JavaScripts to finish loading as well as follow redirects. Furthermore, we present the effects of different cookie blocking options as well as various tracking classification methods, including our own analysis. With the visualization of the gathered data we also present a tracking network of 50,000 websites, clustering redirect tracker in groups.

4.2 Experimental Setup

In order to identify tracking activities, we simulated user browsing behavior within our custom crawler based on OpenWPM [29]. The crawler automatically visits the *Alexa.com* top 50k ranked websites, clicks on a link on these websites (selected by an algorithm described in Section 4.2.4), monitors the web traffic and the events in the browser such as set or deleted cookies. We extended OpenWPM running on Firefox because accessing specific browser functions like HTTP event handlers were easier compared to Chrome. To evaluate the tracking activities we crawled the web for several months between May and October 2018, evaluated the data, improved our crawler and set up our final crawling framework again in December 2019.

4.2.1 OpenWPM Framework

We decided to build our custom crawler based on the OpenWPM framework. In a nutshell, OpenWPM is an Open Source framework written in Python and JavaScript/TypeScript to visit websites automatically. Further, OpenWPM can perform certain actions on those websites and monitor web traffic as well as privacy-related events in the browser. This includes HTTP request and response traffic as well as handling cookies, JavaScript, and other elements.

Technically, OpenWPM consists of a Python script that first configures and starts the browser, then controls it using Selenium. An additional OpenWPM add-on is installed in the browser (Firefox) that monitors HTTP requests, cookies, certain JavaScript APIs, and even more. Upon request, additional add-ons can be loaded in Firefox such as Ghostery to change the behavior of the browser. The gathered data is then saved in a SQLite database.

4.2.2 Crawler Architecture

We enhanced OpenWPM with several features that are useful to study the effect of redirect links: after loading a website, we save a list of all links which are available on that site to our database. We also monitor more JavaScript APIs that might be used by such trackers, like the access to *document.location*. Further, we committed other minor bug fixes and improvements to the OpenWPM Github project, such as collecting the origin of an HTTP request[3], we record if the HTTP channel is being replaced, and store

[3] We added the fields: document_url, top_document_url, referrer_url, and original_url

more details about first/third-party contexts in JavaScript. In addition, we trace if a third-party window is being opend in a full page load in order to identify redirect trackers.

Since OpenWPM does not support distributing the Firefox instances on multiple machines, we implemented a distributed architecture for our crawler based on Celery[4], RabbitMQ[5] and Docker[6]. The crawler architecture is depicted in Figure 4.1. A central server coordinates many crawling nodes through a message queue. The central server queues new tasks that specify which site should be visited with which browser settings. It keeps track of which tasks have already been completed and collects the SQLite files generated by those tasks. It also keeps track of which links were visited with which browser settings so that repeated clicks on the same link with the same browser settings are avoided.

The nodes take jobs from the queue and execute them in individual Docker containers in parallel so that a problem in one of the tasks doesn't affect the other tasks. During the execution of the task, OpenWPM selects the next link to click on in coordination with the master node. The task results are then uploaded to the master node for further analysis.

After the crawl artifacts have been uploaded, the master node imports the individual SQLite files in a joint Postgresql database for further analysis.

4.2.3 Crawler Configuration

We used Firefox with two different cookie settings: *always* which corresponds to the default behavior of Firefox and *never*, which corresponds with the *network.cookie.cookieBehavior=1* setting of Firefox, in which Firefox would only accept first-party cookies. Firefox supports other cookie settings as well, such as turning all third party cookies into session cookies or a mode that isolates third party cookies from different sites. These modes are not interesting for short-running tasks such as the actions we perform. Instead, they are mostly useful for long time usage browsing various websites.

In addition, we disabled popup windows in Firefox since pop windows would result in a new window that loads an additional website. We focus on the events that happen in a single browser window following a link click on that website.

4.2.4 Simulated User Behavior

Our crawler always performs the same task for all browser configurations and a given URL: A fresh instance of Firefox with a fresh profile is started on the requested browser configuration and navigates to the specified URL. It then waits until the page is loaded, then performs a bot mitigation (the mouse pointer is moved around and a mouse scroll down event is performed), which triggers on most websites the loading of additional resources. The script then waits ten more seconds to give JavaScript on that page enough time to execute. Later, all links that currently exist in the DOM are collected and stored in a database. Only <*a*> tags, including <*a*> surrounding <*img*> tags are considered to be a link. Other HTML tags such as <*div*> or *onClick* JavaScript triggered events are not considered to be a link.

In order to achieve a realistic user behavior we used the following methodology to generate a similar URL/Link selection as in the real world page view by AOL users [268]:

1. Select visible links. This prevents clicking links that only exist in the DOM and are only visible when the mouse is hovering on the navigation bar.

2. Prioritize links pointing to a domain that had not been visited before. This is done to avoid visiting popular domains to often, which is unlikely to produce useful data.

[4] Software for asynchronous task queues `http://www.celeryproject.org/`

[5] Open source message brocker software `https://www.rabbitmq.com/`

[6] OS-level visualization to deliver software in packages `https://www.docker.com/`

3. When all available links are equal in this regard, we prioritize links that point to a different subdomain which had not been visited before.

4. We then selected all other not visited links.

5. We decided to implement a biased link selection instead of a random one because our pre-study showed that a random selection resulted in visiting Twitter, Google, and Facebook links in 30% of all crawled pages, since those links are placed more often on websites.

6. After clicking a link we wait until the new page is loaded, perform another bot mitigation, wait for a few seconds, save the HTTP trace, and then close the session.

We click only external links since our pre-study showed more promising data on finding redirects and also to reduce crawling time. To open the same link in all browser configurations, the central link controller (4) selects first the links that had been visited by another browser configuration first. During a crawl, we aim at visiting links only once for each browser configuration.

For our final crawl, we tried to visit up to 34 different links on each website in the Alexa top 50k with the two browser configurations. When a website would not have any more links that were not visited yet in the specific browser configuration, we visited the site again up to 4 times to check for dynamically updated and unique links to visit.

4.3 Results

Evaluating the HTTP traces in our crawls, we notice that 95,8% (tracking) redirects occur by clicking on external links when visiting websites. In order to reduce the crawling time to about 33 days for 50,000 websites and to improve the results, we focused our redirect link evaluation on external links. In total, we ran 1,259,568 individual crawls on the 50k Alexa top-ranked websites using two different browser configurations. During the crawls we clicked in total on 953,603 links (56% embedded on HTML image tags) and encountered 108,435 different redirect chains involving at least one additional external domain. Figure 4.2 displays the number of external links found on websites, indicating that 10% of the websites have no external links and more than half of websites have 4-31 external links.

While only clicking on distinct links, our crawler visited on average 10 external links on each page. Here, all our browser configuration clicked in 99,7% of the cases the same amount of links. Figure 4.3 depicts equal distribution.

Since both browser configurations did not show a significantly different behavior for most aspects of the crawl, except for the handling of third-party cookies, we focus on the default (*always*) configuration of Firefox for the rest of this section and compare it with the *never* configuration wherever needed.

Regarding the term *domain*, we consider everything one level below a public suffix according to the *Public Suffix List* a domain for the analysis, since a cookie can be set for at most an entire domain.

4.3.1 Initial Loading

During the initial loading of a website we already encountered various legitimate redirects. Most websites (77%) upgrade from *http* to *https* using an HTTP redirect, but we saw other types of redirects as well. For example, about 1% of the websites redirect to a country-specific website, such as from *company.com* to *company.fr*, changing the top-level domain but leaving the rest of the domain unchanged. Redirects within the same domain appeared in 25% of our visits, such as a redirect from *www.example.tld* to *example.tld*. For about 5% of our visits, we were redirected to a different domain. We consider every domain that was visited during the initial loading of a website to be the *first-party domain set*.

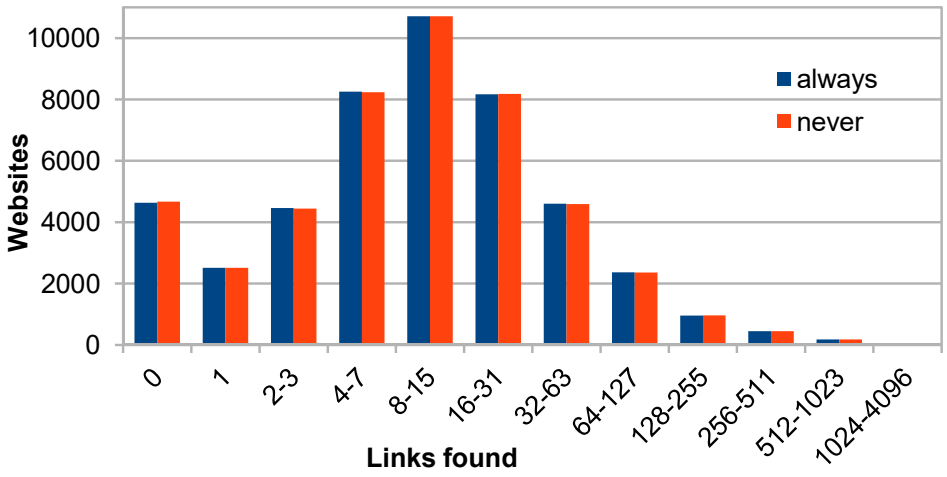

Figure 4.2: Amount of links found on each website (browser cookie config: blue=always, red=never).

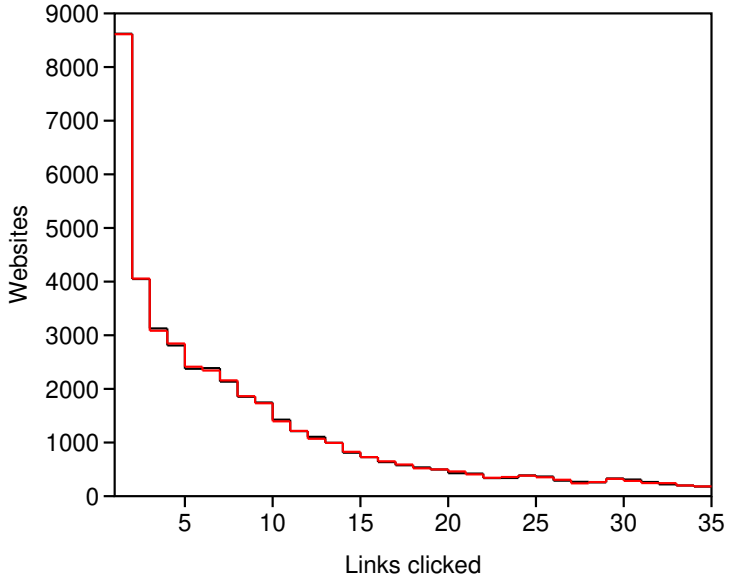

Figure 4.3: Amount of links clicked on websites (nearly similar on browser cookie config.

4.3.2 Identifying Redirect Trackers

To detect redirect trackers, we monitored the URLs that were loaded as top window document and their respective domains. We consider this sequence to be a redirect chain, when URLs from more than one domain were loaded. All elements except for the last domain in that chain are candidates for redirect trackers.However, we did not take domains into account that were already part of the *first-party domain set* from the initial loading of the website, because those domains were already first-party domains and can for example set legitimate first-party cookies. We consider the remaining domains to be the *middle domains* in a redirect chain.

Identifying the most common redirect domains, we kept track of the *middle domains* that occurred in redirect chains starting from the crawled Alexa websites. We built a top 100 list of these domains and then classified them manually. Not every domain on that list performs redirect tracking, some provide other types of services as well. In general, we found the following type of services:

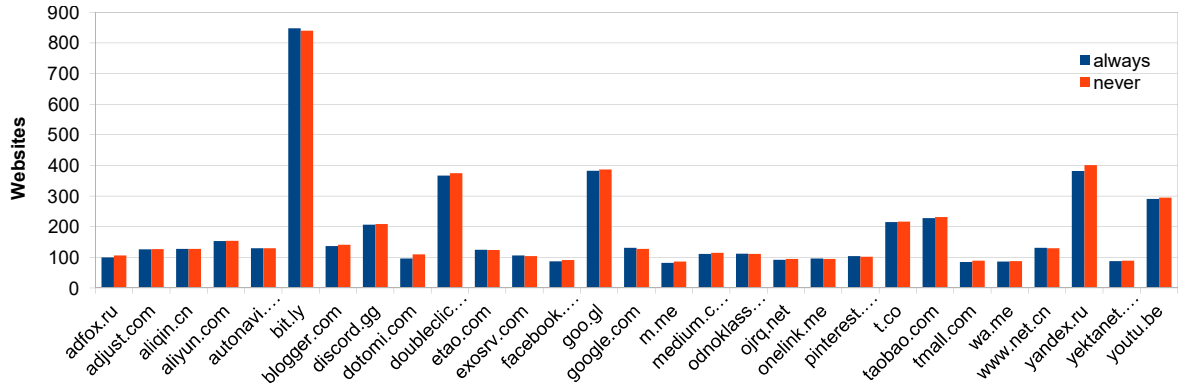

Figure 4.4: Top redirect occurrence on different browser settings.

a) Public URL shortening services such as *bit.ly* or *goo.gl*[7].

b) Privately used URL shortening services that only link to a specific service such as *youtu.be* or *amzn.to*.

c) Privately used URL shortening services such as *t.co* which is used internally by Twitter to shorten URLs in tweets.

d) Traditional click counting services for advertisements or for general websites[8].

e) Popup-Advertisements[9].

f) Other services that for example redirect a user to a social network (*share*) page.

4.3.3 Regular Trackers

In addition to the redirect domains, we evaluated the storing of third-party cookies on different websites during the initial page load. For the domains that accrued most often, we computed their coverage of the overall Alexa top 50k, presented in Table 4.1. This shows the potential that a tracker can re-identify the users browsing behavior.

Table 4.2 lists the 30 most common redirect domains we found with our classifications. The column *Sites* indicates on how many distinct websites we found at least one link that led to a redirect chain containing this domain as part of the *middle domains*. In general, cases *d)* and *f)* of Section 4.3.2 are difficult to distinguishs since they sometimes interact with each other. For example a new popup window is created and the user is redirected over several domains till he reached his final destination. About 11.6% of the websites on the Alexa top 50,000 had at least one link leading to one of the top 100 redirectors and about 8% had at least one link that lead to one of the top 30 redirectors shown in Table 4.2.

Noteworthy are the redirect trackers *yandex.ru*, *doubleclick.net*, *google.com* and *facebook.com* that also cover a significant part of the Alex top 50k domains (see Table 4.1 for a comparison). Besides setting third-party cookies, *doubleclick.net* is also used redirects in 4% of the cases pointing back to the domain the redirect was triggered. This special cased is probably used to set first-party cookies during the redirect and to mitigate ad-blocking plug-ins, that usually block third-party cookies.

[7] Everyone can submit a URL and gets a shortened version of that URL.

[8] A website or advertisement links to the service. When a user clicks on these links, the click is tracked by the referenced server and the user is then redirected to the real destination.

[9] When a user clicks on a website, a new window (popup) is opened which loads a new website. In general, the window loads the URL of the advertiser first and is then redirected to the advertised product. This is quite similar to d) when the user has popups disabled.

Rank	Domain	Coverage in %
1	doubleclick.net	42
2	facebook.com	26
3	adnxs.com	19
4	casalemedia.com	16
5	openx.net	15
6	everesttech.net	15
7	pubmatic.com	15
8	quantserve.com	14
9	bidswitch.net	10
10	yahoo.com	10
11	agkn.com	10
12	addthis.com	10
13	mathtag.com	9
14	adform.net	9
15	atdmt.com	9
16	google.com	9
17	adsrvr.org	9
18	teads.tv	8
19	scorecardresearch.com	8
20	turn.com	8
21	dnacdn.net	8
22	mookie1.com	8
23	bing.com	7
24	smartadserver.com	7
25	demdex.net	7
26	taboola.com	7
27	simpli.fi	7
28	spotxchange.com	7
29	innovid.com	7
30	nr-data.net	7

Table 4.1: Top 30 regular trackers encountered in our crawl.

4.3.4 Redirect Trackers and Cookies

Not all redirectors track the user with cookies. For example, *youtu.be* is a URL shortener for YouTube and directs only towards *youtube.com*. It never sets a cookie, which is also not required since the next hop *youtube.com* sets cookies to track users. Also *goo.gl*, which is a URL shortening service operated by Google does not track users and never sets a cookie during a visit.

To generalize the use of cookies, we classified the use of cookies in three categories:

NV No cookie is set by the redirect domain.

AP While the initial page loads, the redirect domain already sets a (third-party) cookie in the browser. This happens due to third-party content from this domain being included during the initial page load as well.

AC No cookie is set from the redirect domain during the initial page load, but after the link is clicked and the browser follows the redirect chain, a cookie is set.

Rank	Domain	Class	Redirects	Sites	config always			config never			TC	SR	Alexa	EL	EP	GH	ITP
					NV	AP	AC	NV	AP	AC							
1	bit.ly	a	1941	848	0	1	99	0	0	100	100	242	0			x	x
2	goo.gl	a	593	382	100	0	0	100	0	0	0	55	0			x	x
3	yandex.ru	d	5150	381	0	100	0	0	0	100	100	4	5		x	x	x
4	doubleclick.net	d	2362	367	0	98	2	4	0	96	100	97	42	x	x		x
5	youtu.be	b	539	291	100	0	0	100	0	0	0	0	0			x	x
6	t.co	c	1244	215	0	0	100	0	0	100	100	101	0			x	x
7	discord.gg	b	225	207	0	0	100	0	0	100	0	0	0			x	
8	aliyun.com	d	159	152	4	1	95	6	0	94	1	1	0			x	x
9	www.net.cn	d	131	131	100	0	0	100	0	0	0	0	0				
10	autonavi.com	f	130	130	100	0	0	100	0	0	0	0	0			x	x
11	blogger.com	f	797	128	90	10	0	100	0	0	10	1	0			x	
12	aliqin.cn	d	128	128	100	0	0	100	0	0	0	122	0			x	x
13	google.com	d	565	127	27	40	32	35	0	65	72	24	9	x	x	x	x
14	adjust.com	d	260	126	0	0	100	0	0	100	0	28	0	x			x
15	odnoklassniki.ru	f	133	112	100	0	0	100	0	0	0	0	0			x	x
16	medium.com	f	160	110	0	1	99	0	0	100	100	86	0			x	x
17	exosrv.com	d	808	106	0	95	5	0	0	100	0	0	1				
18	pinterest.com	f	106	104	0	7	93	100	0	0	6	0	1		x	x	x
19	taobao.com	d	552	102	0	7	93	0	0	100	76	3	0		x	x	x
20	adfox.ru	d	755	100	51	47	3	100	0	0	49	58	0		x		
21	onelink.me	d	155	96	1	6	93	3	0	97	99	31	0			x	x
22	dotomi.com	d	181	96	0	1	99	0	0	100	100	0	0	x	x	x	x
23	ojrq.net	d	181	92	0	2	98	0	0	100	35	1	0		x	x	
24	yektanet.com	d	1656	88	0	95	5	0	0	100	99	1	1		x		x
25	facebook.com	f	101	86	45	53	2	100	0	0	0	4	26		x		
26	wa.me	f	1030	86	100	0	0	100	0	0	0	0	0				x
27	m.me	f	960	82	100	0	0	100	0	0	0	0	0				
28	app.link	d	230	78	0	47	53	0	0	100	100	25	0		x	x	x
29	securecloud-smart.com	d	96	78	0	0	100	0	0	100	0	0	0				
30	buysellads.com	d	1072	77	0	0	100	0	0	100	0	0	0				x

Table 4.2: Top redirectors and their properties. Class: Describes the redirect classification. Redirects: How many redirect chains we observed that included this redirector. Sites: On how many different websites we observed this redirector. The config always and config never columns show how many times (in percent) a cookie was set from this domain. NV: Cookie was never set, AP: Cookie was set after the initial page load, AC: Cookie was set after the click on a link during the following redirect chain, TC: Percentage of redirects in which the redirector set a tracking cookie, SR: Number of self redirects observed for this redirect pointing back to the original website the redirect link was clicked on. Alexa: Percentage of the Alexa top 50k that had third-party cookies from this domain, EL: Domain is on the Easylist, EP: Domain is on the EasyPrivacy list, GH: Domain sets cookies within Ghostery plugin, ITP: Domain sets cookies within Safari ITP 2.3 (release March 2020).

Not all encountered redirectors behave similarly. For example *yandex.ru* operates different services such as advertisement and posting links to social media feeds under the same domain. Also, *app.link* uses different subdomains for different customers and some customers embed additional tracking script as well while others just include a link towards *app.link*. Depending on which service is used and how the service is used, the redirector sets a cookie during the initial page load (AP). Table 4.2 shows an overview of how often (in percent of the redirect chains) we encountered each behavior for the default cookie settings (*always*) of Firefox as well as for the more restrictive *never* setting that prevents third-party domains from setting cookies.

Besides the cookies that were set in our crawl, we also compared the redirector domains we encountered with the EasyList and the EasyPrivacy list[10], which are well-known lists of domains that perform tracking or deliver advertisements. While most of the domains that serve advertisements and set cookies are included in one of the lists, a few such as *exosrv.com* are not classified as tracking or advertisement. Similar, URL shortening services that set cookies such as *bit.ly* or *t.co* are classified as trackers by EasyList.

In addition, we evaluated the top 30 redirect tracker domains within the Ghostery plug-in of Firefox as well as the recent release of Safari ITP 2.3 (March 2020) [269]. Both claim to block tracking cookies also during redirection. Since we could not automate the analysis process with Safari, we manually opened and clicked the same crawled websites as well as redirect links extracted from the database. Our analysis visualized in Table 4.2 shows, Ghostery could only block 11 out of 30 top redirect domain cookies, whereas Safari blocked 9 out of 30. Here, Ghostery removed in 5 out of 11 blocked test cases the complete redirect link, not being able to start the redirect process. While Safari was blocking 9 redirect cookie domains, those cookies were still present in the browser cache. Moreover, different other cookies have been set in all test cases during the redirect process, showing that only a portion of the known trackers are blocked within Ghostery as well as Safari. Notice, during our pre-studies as well as the manual evaluation we did not notice any CAPTCHA anomalies, which is the reason we did not look into this topic in more detail.

To check whether the appearance of specific redirector domains changes when we modify our cookie settings in the browser, we compared the top 30 redirectors for our two browser configurations. Figure 4.4 displays the results and shows that using more restrictive cookie settings does not affect to which redirector we are directed to.

4.3.5 Cookie classification

Not every cookie is a tracking cookie. In general, a cookie is a tracking cookie when it is used to track the activity of a user. This is hard to determine since this can only be determined with knowledge about how the cookie is processed on the server. However, tracking a user with a cookie is only possible when *(1)* the cookie has sufficient entropy so that it can be distinguished from cookies of other users and *(2)* when the cookie has a sufficient lifetime and is not just a session cookie. Other properties of the cookie might give a hint whether it is used as a tracking cookie or for other purposes as well.

We decided to implement a cookie classification algorithm similar to the algorithm used in [270, 29] with a few modifications. We consider a cookie to be tracking cookie when:

- It is not a session cookie and the lifetime is at least 90 days.
- The length of the cookie value is at least 8 bytes.
- The length of the different values observed during our crawl differs by at most 25%.
- It is user-specific, so every value for this cookie is only seen in a single visit.

[10] `https://easylist.to/`

- The similarity of the value compared to other values of the same cookie according to the Ratcliff/Obershelp string comparison algorithm[271] is 66% or less for at least half of the other values we observed in our other visits.

The original algorithm suggested checking for a constant cookie value length. However, we found cookie values encoded using *url encoding* so that the length of the cookie value changed a lot, while the underlying data changed only slightly. Furthermore, we found integers in the cookie values encoded without leading zeros for small integers, which had an additional effect on the length. Manual investigation on those examples showed that those cookie might be used for tracking, which is why we modified the tracking classification algorithm as described above. The result of the automatic cookie classification can be found in Table 4.2 in the *TC* column. While we believe this is mostly accurate, there are a few unclear cases such as *exosrv.com*, with *url encoded* cookie values. The cookie is an encoded data structure and some elements of the structure might be used as identifiers for tracking as well[11].

4.3.6 Clusters of Trackers

Since redirect trackers often appear in chains with more than a single tracker in the chain, we decided to take a look at the interplay between the 100 most common redirectors as well. We considered the length of the sequence of all domains we encountered after clicking the link before we reached the final destination with consecutive identical domains only represented as a single entry to be the length of the chain. For example, when we click a link on *a.com* and then go to *b.com/a*, *b.com/b*, *c.com*, and then finally arrive at *target.com*, this sequence has length 2, since *b.com*, *b.com*, and *c.com* are the middle domains, but since *b.com* appears twice, we remove the second appearance and just consider the sequence *b.com*, *c.com*, which has length 2. Table 4.3 shows the distribution of the lengths of the chains we encountered according to this definition.

Number of redirects	41414	7970	3432	771	288	114	32	10	5	3	6	6
Chain length	1	2	3	4	5	6	7	8	9	10	11	≥12

Table 4.3: Redirect chain length (chain length 1 indicates only a single redirector appeared between start and target).

To understand this behavior, we build a directed graph of connected redirect domains shown in Figure 4.5. We also created a total graph with all the 9,862 redirectors we encountered in our crawl, but only used to compute metrics shown in Table 4.4. Due to the high number of nodes it's not visualized here. The nodes in our graph are the redirectors that appeared most often in a redirect chain. An edge is drawn from node A to node B when node A redirected the user to node B during our crawl. The edges are weighted by the number of redirects we observed from node A to node B during our crawl. Only redirects within a redirect chain are considered here. The more connection exists, the darker shade of red and the thicker the edge is plotted (e. g., *buysellads.com* had 507 connections to *doubleclick.net* - shown in the bottom of the figure). The nodes size depicts the out-degree of a redirect domain. For example *doubleclick.net* has an out-degree of 22 and *bit.ly* 21 outgoing connection.

We then used the Louvain community detection method [272] to cluster the redirectors. Each cluster is represented in a different color. The ten main communities contain between 3 (red color) and 21 (purple) nodes. One of the biggest clusters forms around *bit.ly*, connecting to 21 nodes in the top 100 graph and to 167 other domains (in and out-going connection) in the total graph, including e. g., *smartadsrv.com*, *goo.gl*, *amazon-adsystem.com*, *servedbyadbutler.com*, *youtube.be*. Another major cluster exists around *doubleclick.com*, connecting to 22 nodes in the top 100 graph and 84 in the total graph,

[11] Sample cookie exosrv.com: *a%3A1%3A%7Bi%3A0%3Bs %3A33%3A%225e01e859d10810.010044542057158735%22%3B*

Graph	Total	Top 100
Weakly connected components	606	25
Strongly connected components	3534	79
Modularity	0.81	0.83
Average clustering coefficient	0.019	0.072
Average path length	6.027	3.273

Table 4.4: Redirect graph statistic.

```
https://share.yandex.net/go.xml?service=gplus\&url=http\%3A\%2F\%2Fkino-history.
net\%2F\&title=Kino\%20History\%20-\%20\%D1\%81\%D0\%B0\%D0\%B9\%D1\%82\%20\%D0
\%B8\%D1\%81\%D1\%82\%D0\%BE\%D1\%80\%D0\%B8\%D1\%87\%D0\%B5\%D1\%81\%D0\%BA\
%D0\%B8\%D1\%85\%20\%D1\%84\%D0\%B8\%D0\%BB\%D1\%8C\%D0\%BC\%D0\%BE\%D0\%B2\%20
\%D0\%BE\%D0\%BD\%D0\%BB\%D0\%B0\%D0\%B9\%D0\%BD
```

Listing 4.1: The URL is used to direct the user to Google+ so that he can share a posting about kini-history.net there. While it points to share.yandex.net, the real destination can be easily concluded from the URL.

including e.g., *buysellads.com*, *carbonads.net*, *smartadserver.com*, *bit.ly*, and others. Table 4.4 gives an insight into the graph statistics of the total redirect connection graph as well as the filtered top 100 graph. We see that the top 100 graph has a high number of strongly connected nodes while the total graph is less connected and has only a clustering coefficient of 0,019.

We can see that users are often directed to *bit.ly* in a redirect chain from a lot of different origins and URL shortening services often interact with it. Another major cluster appears around *yandex.ru* and *yandex.net* that is connected to many other services from Russia. We can also see that *doubleclick.net*, which is a major platform for advertisements regularly interacts with other smaller advertisement services as well. Some redirectors are not connected with other nodes in the graph at all.

4.3.7 Structure of the Redirect URLs

While we managed to locate many redirect trackers, they often share a common structure in their URLs. In general, we encountered the following cases:

Fixed destination, target easy visible For some URLs, the real target is easily visible when looking at the URL. An example for such URLs can be found in the Listing 4.1. We think that such URLs can be easily rewritten to their real target using regular expressions.

Fixed destination, target hard not obvious Some other URLs redirect the user always to the same page, but the redirection target is not easily visible. An example of such an URL is given in Listing 4.2.

Fixed destination, target known to server This is typical for the URL shortener. For example `https://bit.ly/2AGeopq` redirects the user to `https://petsymposium.org`. However, only the server knows that this is the destination for this URL.

Random destination This is often encountered for advertisements. The URL just points to a server, which then decides based on the users tracking cookie, the IP address and probably many other factors to which page the user is then redirected to.

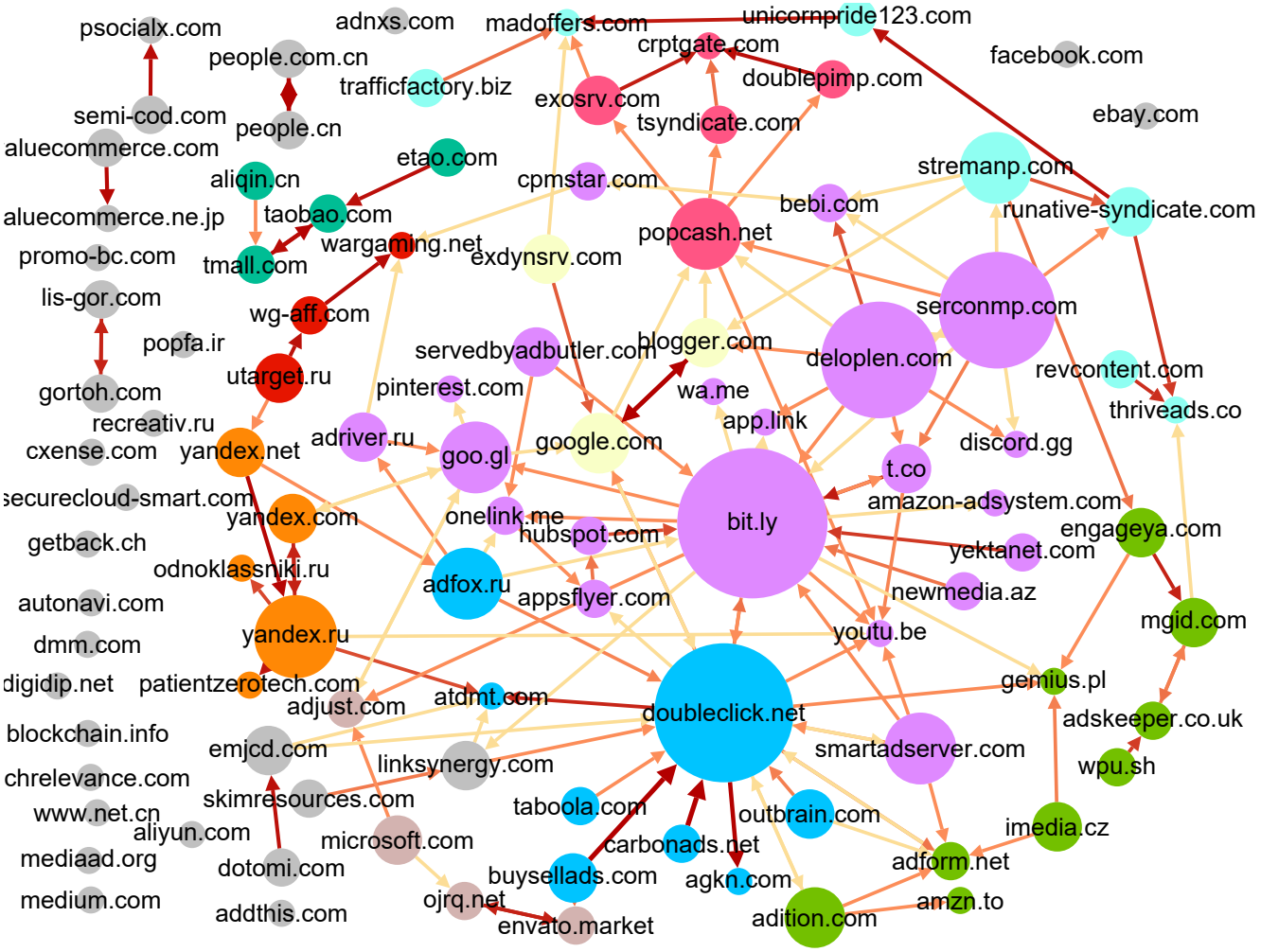

Figure 4.5: Dependencies between the redirectors. The width of an arrow from a to b shows how often we have seen a redirect from a to b in our crawls.

```
https://clck.yandex.ru/redir/dRu0iBr5YqxShfdSGFXg7CqFK-IM7KfY?data=eE03aEhZNGdu
WC1aR1BqRE9uSW1PYUdtOVMzbzV2eUdDYmhFOHZCbjIwZkpJZTk1TU1yRzNMS0VIWWdJOW5keGtrU3R
paXJlcUdVMGJ5QWxiT05sYzV3QjhhUDFESzVBSlRMQTFrcEZPWFNFWkFtNm90WElIOVAxVnBTMHJuLW
VOYlFZTXFTazFQdW9nUmdMLTltM0lmd1hUWkppN0s2OHB1LTB2TVMyVjBDaklfR2xVYWo5TDBnbWV4U
nJuMmNYTVFkY3B3V1dIZlh2Z01abFZ0c1AxaGFvaXZaOTlvVDgtX1NiZXlTbldEVWdGNWlOZXlBanFk
ODhTS0RGWFh4SXl6LWlUV2JwRUppb25vRXR3\&b64e=2\&sign=489b163e63fdd8d4ad5bc56de4b5
be68\&keyno=3
```

Listing 4.2: The URL directs the user to Facebook to post something about *kino-filmi.net* on his timeline. However, from the URL itself, it is less obvious that this is the target of this redirect.

4.4 Countermeasures

Since blocking third-party cookies might become more common, we assume that tracking through redirects might become more common too. To counter redirect tracking, we propose the following countermeasures.

4.4.1 Rewriting URLs in the Browser

There are a few redirectors for which the destination URL can be easily predicted from the URL parameters, we recommend to build rules for the most prominent redirect trackers that rewrite the requests directly to the destination server. The redirecting hop is then skipped and therefore no cookie is set. The tracker doesn't even know whether the URL was assessed or not. A similar approach is used by many ad-blocking add-ons, that keep lists of URLs that should be blacklisted. The disadvantage here is that those lists need to be maintained and less prominent redirect trackers might not appear in those lists.

4.4.2 Blocking Cookies from 1st Party Redirects

The redirect trackers for which the URL cannot be automatically rewritten, we have to contact the redirector to get the final destination. Yet, we can block all cookies from those hosts. Safari has already implemented a similar procedure [273], but lacks in many cases as our analysis shows. Nevertheless, it might be possible that redirect trackers will then move to other redirection techniques such as loading a website that then redirects the user using JavaScript. Our crawler is based on Firefox 70, which currently provides no protection against redirect tracking. In general, we think that it is a good idea to turn cookie from domains that were a first-party just for an HTTP redirect into session cookies with a short lifetime or to block them in general. Since redirects can also be performed using HTML meta tags or JavaScript, one may require user interaction here as well as Safari does.

A few redirect trackers we encountered are also known to ad-blocking or privacy-enhancing add-ons such as Ghostery. Previous versions of Ghostery blocked some redirectors so that instead of being redirected to the final destination of the chain, an error message was displayed in the browser window. Since this harms the user experience, this was disabled recently by Ghostery and the user is redirected through the redirect chain to the final destination, but the cookies of those domains are blocked. This works in most of the cases as our analysis shows.

Another effective technique is to automatically isolate the locally stored website state (including cookies, LocalStorage, and browser cache) into separate containers [23]. As shown in the large scale study, the isolation concept reduces the number of pages tracked by 44%. Mozilla has been working on a similar concept called First Party Isolation, separating cookies on a per-domain basis, for several years. Originated in the Tor Project, where it was known as Cross-Origin Identifier Unlinkability (double-keying)

[274], it was added to Firefox 52 as part of the Tor Uplift project [275, 276]. Since Firefox 77, the so-called Dynamic First Party Isolation feature was made available to configure the isolation of cookies through the user interface [277]. As discussed in [23], a reason why it was not enabled by default by Mozilla, is that it may break some websites when activated.

4.4.3 Blocking Popups

We disabled popups for our crawl since we wanted to focus on the events in a single browser window. However, we also noticed that popups are sometimes combined with redirectors and when a new popup window is created, it is first directed to a chain of redirectors before the content of that window is loaded. Disabling popups is another way to reduce the number of redirectors encountered while browsing the web.

4.4.4 Blocking Ads

A significant part of the top 30 redirectors we encountered comes from clicks on advertisements (mostly on images, which make 56% of redirect links). By simply filtering advertisements, like ad-blocking plug-ins such as Ghostery are doing, those links will not show up in the browser so no user is not able to click on them. However, not all redirect image links are ads, other categories are for example social media icons or news pictures, which are difficult to distinguish. In addition, as our manual evaluation on the top 30 redirect trackers showed, there also exist text links to further content such as news or suggested page content. Filtering out those links is also difficult.

4.4.5 Browser Add-on LinkTrackExchange

We developed the proof of concept browser tool *LinkTrackExchange* to mitigate redirect link tracking. The general concept is replacing (redirect tracking) links inside a visited website with their final URL, striping out redirect tracking chains. *LinkTrackExchange* is based on a client/server architecture similar to ad-blocking tools. By using our crawling architecture, the server continuously crawls a given and updated list of (Alexa) domain URLs in order to follow all links inside the website to identifying redirect chains. In addition, the server is storing the clicked links together with the final link destination URL in a SQL database.

The client-side browser add-on then queries the server for each visited website if any (redirect tracking) links on that website needs to be replaced in the DOM by the final destination URL (stripping out all redirect tracking chains).

Implementation: On the client-side, we implemented *LinkTrackExchange* as a Greasemonkey[12] user-script. Greasemonkey is a Firefox extension that allows users to install scripts customizing a website on-the-fly. The script can also be installed on different browsers supporting user-scripts such as Chrome, Safari, or Opera by using built-in features or extensions such as Greasemonkey or Tampermonkey[13]. We chose Greasemonkey due to the multi-browser compatibility.

LinkTrackExchange is using the Document Object Model (DOM) of a website to exchange all redirect tracking links before the website is displayed to the user. Running in the browser's background, *LinkTrackExchange* waits until the DOM of the requested website has been build and the *DOMContentLoaded* event is fired. At that point the in-line dependencies like scripts have been loaded.

Process: In the initialization process *LinkTrackExchange* checks if the latest version of the redirect database is already saved locally at the client's machine or was saved within a predefined time (default is 24 hours). If no database is available or is older then the predefined time, *LinkTrackExchange* opens

[12] `http://www.greasespot.net`
[13] `http://tampermonkey.net`

an *XMLHttpRequest* (XHR) to the database server. A server-side PHP script triggers a SQL query to obtain the requested redirect data from the database. The server then sends the database output back to the browser formatted in JSON. Next, *LinkTrackExchange* parses the data containing the redirect links for every redirect-tracking website. The data is saved in the local SQLite database. This caching mechanism helps to increase the performance of exchanging the redirect links and to preserve the users browsing privacy by not querying the server on every visited website. Once *LinkTrackExchange* is up to date, the tool verifies if the requested website is saved in the local database and therewith contains redirect links. On a positive match, *LinkTrackExchange* scans all <a> tag elements to replace the redirect links with the final destination URL saved in the database. If the website is not in the database, the script sends the host URL together with a random list of 10 other URLs out of the saved URL list to our scanning server using additional random URLs within the request will give an additional privacy feature to the user, obfuscating the visited and unknown URL. If the domain was not already in the scanned list, a PHP script adds this URL to the list of websites that needs to be scanned in the next crawling process.

Limitations: If a link contains parameters that are generated on every request, link replacement problem could accrue. By using machine learning algorithms we plan to predict the correct parameters in future releases. However, as our analysis shows, tracking IDs are the frequently changing part of redirect links. By stripping out those parameters we enable an additional privacy feature to the user. If a website is dynamically changing the redirect links after a page load, we plan to intercept those changes during page load.

Mobile usage: When using the redirect protection tool on mobile browsers (smart-phones), we do not download the entire redirect database on the user's device. This saves network bandwidth and storage capacity. Instead, when a user requests a website, *LinkTrackExchange* queries our redirect server (together with a random URL list) if any redirect links for those hosts exist. On a positive match, the redirect server sends all redirect links with the corresponding target links to be exchanged back to the client. *LinkTrackExchange* then parses all <a> tag elements, to replaces all redirect links with the final destination URLs.

For future work we will evaluate the performance of a proxy server to process all requests and to exchange redirect tracking links on-the-fly instead of the local script.

Figure 4.6: Top 100 publishers, colored/sized by tracker intensity.

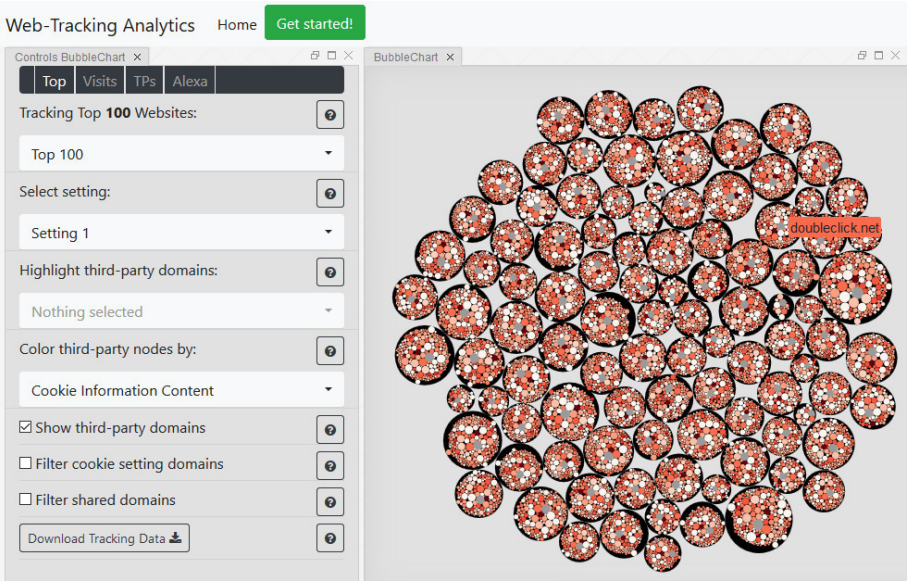

Figure 4.7: Top 100 publishers showing embedded tracker.

4.5 Visualization

Presenting the massive tracking potential of visited websites (publisher) of a single crawl we implemented various interactive web based visualizations. Each visualization allows to gather a specific perspective on the dataset. The visualizations of our data are realized with the popular JavaScript library for interactive data-driven visualizations D3.js [278]. The visualization is not just limited to redirect tracking, but shows a general overview of all kind of tracking elements such as tracking pixels, external JavaScript or iFrames embedded into the website.

4.5.1 Bubblechart

Within the Bubblechart the user can select various options to visualize the appearance of third-party and tracking elements found on the crawled websites. The intensity of classified tracking elements is displayed by the size and color of the bubbles representing a domain. A bigger size and darker red color represents an intense tracking activity. No tracking activity is represented in white. A publisher embedding more tracking elements and a tracker with a higher cookie information content is colored in a darker red hue. Using additional configuration settings it is possible to highlight the appearance of specific trackers, visualize the effect of different tracking classification methods, or the effect when using different third-party cookie settings as well as the ad-blocker plug-in Ghostery within the browser. For example, in the Easylist and OpenWPM tracking classification we only use red for tracking and a white color for no tracking, since the classification is binary.

The Bubblechart visualizes the publisher with its third-party and tracking elements in four representations (Top, Visit, Alexa and TP) the user can choose from (Figure 4.7).

TOP: Represents publisher that include the top most trackers. The tool displays the top X (10 - 200) ranked websites (publisher), that include the most tracking elements. A tracker is defined by the selected tracker classification (cookie information content, calculated by Shannon's information entropy / Easylist / Extended Easylist / OpenWPM).

Visit: Displays the custom selected publishers and their tracking elements.

Alexa: Shows the publishers that are most visited by users (ranked by Alexa.com).

TP: Displays all websites interacting with the selected third-party domains.

Configure display settings: The following settings can be adjusted to change the view.

- **Select settings:** Display/hide tracking elements according to the following browser settings: *"save third-party cookie" setting: Always, Only from visited, Never / Activate Ghostery plug-in and cookie setting always*.
- **Highlight third-party domains:** Highlights all websites that are interacting with the selected third-party domains.
- **Color third-party nodes by:** Publisher and third-parties are colored according to four different tracking classification methods: *Cookie information content, Easylist, extended Easylist, OpenWPM*.
- **Show third-party domains:** Toggle showing all third-party elements, that are involved when visiting a publisher. Size of the bubble is mapped to the percentage of a domain's requests among all third-party requests that happened during the visit.
- **Filter cookie setting domains:** Only show those third-party domains that are setting cookies.
- **Filter session cookies:** Only show tracking domains, that do not use session cookies.
- **Filter shared domains:** Only show those third party domains which are interacting with all visible websites.

4.5.2 Redirect-Graph

Selecting specific publishers (bubble), the redirect-graphs visualize the appearance of redirect tracker within the different browser cookie/Ghostery settings captured during the crawl. The interactive graph highlights the tracking domains while hovering over the bubbles and shows how many redirect hops happen.

Filter options:

- **Aggregate breadth:** Aggregate the redirect trees by breadth. All nodes of the same domain of each depth are aggregated to one node.
- **Aggregate depth:** Aggregate the redirect trees by depth. Consecutive nodes with the same domain get aggregated into one domain, e.g. google.com–google.com–ad.com–google.com will be aggregated to google.com–ad.com–google.com.
- **Highlight visit domain:** Highlights all redirects that were going back to the website the redirect started from.
- **Toggle columns:** Toggle the columns of the visualization between showing all four or just the column.

4.5.3 Select different statistic data

Statistic data comparing the cookie saving settings (*always/never/from-visited*) on the amount of first- and third-party cookies, HTTP and JavaScript cookies as well as external links is displayed in histograms (Figure 4.8, bottom). The histogram shows the distribution of embedded third-parties within the visited publishers as well as showing the trackers with the highest tracking cookie information content.

- **Count:** Show the amount of first- and third-party requests, profile cookies, JavaScript cookies, found links, and clicked links that are detected in the data set.

- **Top:**
  - *Third-Party Embedder:* The websites with the most embedded third-parties.
  - *Tracking Embedder:* The publisher embedding the most trackers (tracker = cookie information content > 72).
  - *Tracker:* The tracker with the highest information content. Third-Parties present in less than 10 websites are ignored.
- **Third-Parties:** The third-parties present in the most websites.
  - *Entropy Dist.:* A histogram with 10 bins, binning the cookie information content.
  - *Tracker Dist.:* A histogram with 10 bins, binning the number of trackers in a website. Tracker = cookie information content > 72; For example, in bin 1 are 3564 websites that contain 0-5 trackers.

4.5.4 Network-Graph

The network graphs depicted in Figure 4.9 displays the interconnection between publishers and trackers as well as a separate redirect network structure we discovered during our crawl. The interactive graphs visualize most dominant tracker as well as publisher by color and size. Hovering on each node/bubble, the tool highlights the connections to each tracker or publisher and shows individual clusters of connecting nodes. In the redirect network graph we also set a darker color as well as bigger size on the edges (connections) between the trackers that have higher interactions, showing more dominant players.

- Alexa top 200 websites.
- Redirects found during the crawl: Visualizing interaction by linking connections/redirects between publisher and tracker. Highlighting selected connection while hovering over the nodes.

Additional Settings:

- Cookie saving options (*always, never, only when visited, Ghostery plug-in*)
- Color coding by tracker classification (*cookie information content, Easylist, extended Easylist, OpenWPM*)
- Pruning nodes by degree (2-20)

4.6 Summary

We studied tracking techniques hidden behind link redirection. Our evaluation shows that 11.6% of the Alexa top 50k ranked websites contain links that lead the user to the final destination over one of the 100 most common redirectors. We demonstrate that many of those redirectors set (tracking) cookies even when the browser is configured to accept first-party cookies only. This is because the redirector will become the first-party domain during the redirect. Further, we show that many of the domains that perform redirect tracking also perform regular user tracking and are widespread in the Alexa top 50k. Therefore, redirect trackers have an enormous potential for tracking users and once browser vendors restrict the use of third-party cookies by default in the upcoming browser releases, this might become even more common. Moreover, in our redirect community network graph we show how well-connected the redirect trackers are. This makes sharing the gathered user browsing behavior possible for redirect trackers.

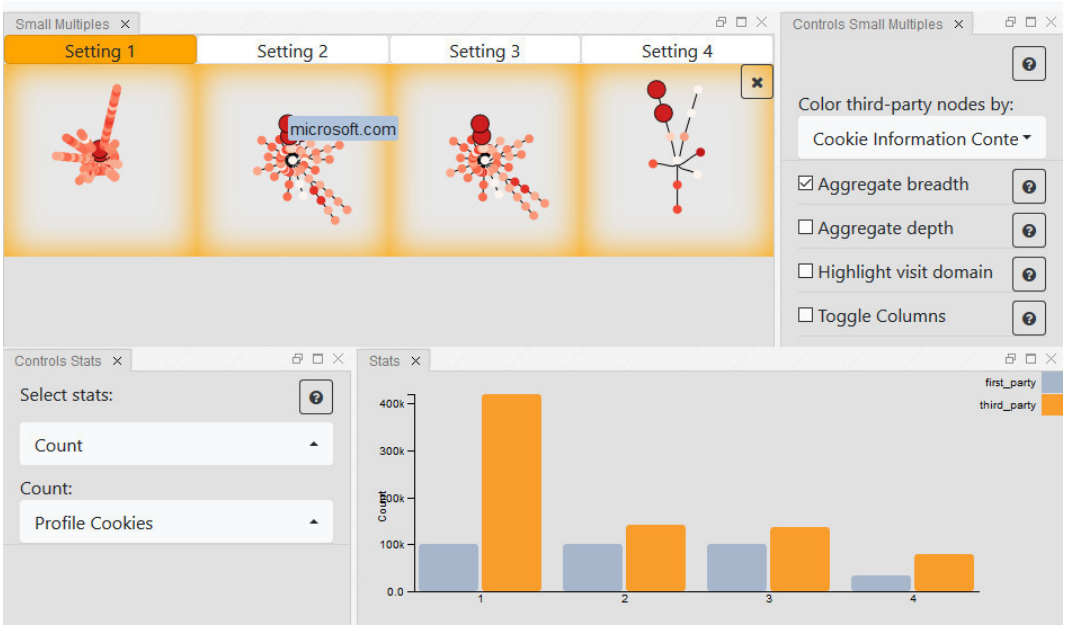

Figure 4.8: Redirect-Graph highlighting 'microsoft.com' and statistic data below.

On the other hand, redirect trackers are rather easy to detect. With our method, it is possible to compile a list of the currently most common redirect trackers on the web and their use of cookies in an automated way. The list can further be refined by a more in-depth analysis of the cookies set by those redirectors, which can be partially automated based on entropy or manual investigation. Current browser add-ons such as Ghostery or ad-blocking add-ons, in general, will already detect some of those redirectors and block their cookies.

There are also positive aspects of redirect trackers. As long as the redirect is performed by HTTP only, it cannot be combined with other techniques such as browser fingerprinting, which requires JavaScript code to be executed to collect all the fingerprinting features.

We see the potential for further research in many directions. First of all, we hope for new and innovative ways of how tracking through redirectors can be mitigated without having to rely on a blacklist. Treating first-party cookies from domains that were only encountered through an HTTP redirect chain in a special way (such as session cookies with a short lifetime) is a good start for that. We would also like to see a long term study on redirect tracking and how it evolves over time. Many of the improvements we have mentioned can probably be implemented using the add-on API of Firefox or Chrome and similar APIs of other browsers. In addition, it will be interesting to see how the current trackers will react to the tracking prevention that was introduced recently in Apple Safari [269] as well as Mozilla Firefox [279].

Finally, it would be nice to have the detection of potentially privacy-invasive redirect trackers included in automatic analysis tools such as for example PrivacyScore[14] so that normal users are made more aware of them.

[14] `https://privacyscore.org/`

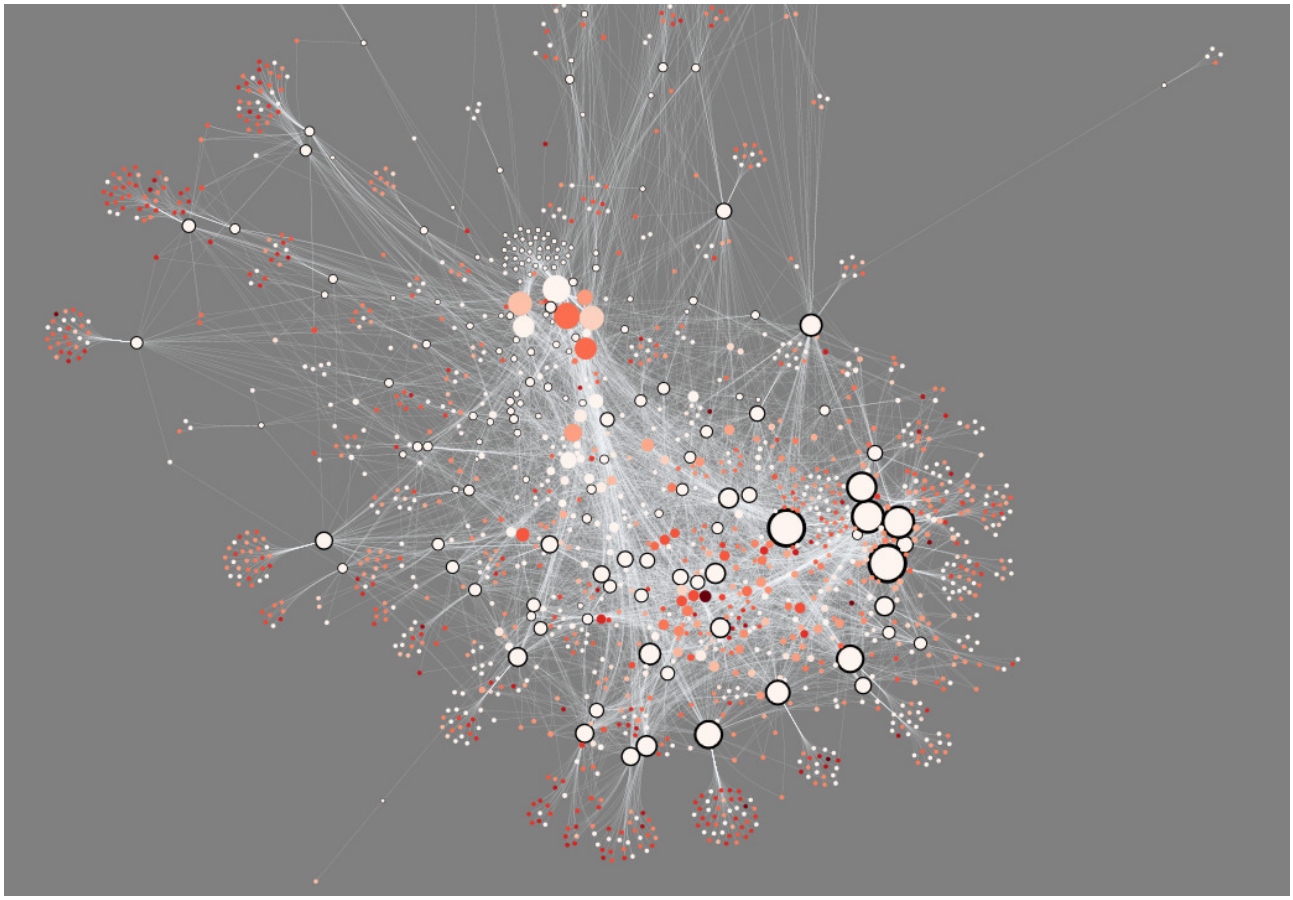

Figure 4.9: Ordered network graph: Publisher colored white with black border connecting to embedded third-party elements colored/sized by tracking intensity.

5 Reducing User Tracking through Automatic Website State Isolation

As research shows [44] and our evaluation in Chapter 3 confirms, the most common way for third-party trackers to identify users is by storing a unique ID on the client and retrieving it on every page the tracking code is embedded. The most basic method to save this ID is by using an HTTP cookie. Any HTTP request suffices for this, but a tracking pixel, a transparent 1:1 pixel sized image, is the traditional, bandwidth-conserving, and high-performance way of doing it. Once implemented, the visitor's browser will request the tracking cookie each time a page is loaded, because caching this image is forbidden by the server. After the first request where the cookie is set, every further request carries the (third-party) cookie and the tracking server can re-identify a user on every page that includes the server's tracking pixel. Gathering more data on visited websites allows more detailed user profiles to be built.

To avoid a customized ID being saved by the trackers, users can disable all mechanisms to save data. This will eliminate tracking based on storing custom identifiers, but cannot protect against using pre-existing data, i.e., browser fingerprinting. More importantly, it affects every site that tries to use the saved data for *any* purpose such as an user login. Since nearly every interactive website uses cookies to save session data, this is hardly practical. A less restrictive method is to disable only third-party cookies. Third-party cookies are cookies being set (or retrieved) from a different domain than the one in the browser's address bar. If a cookie is considered first-party or third-party therefore depends not only on the cookie itself, but also the currently opened page. When visiting *a.com* a tracking pixel tries to set a cookie for *b.com*, this cookie is considered third-party and is ignored. Ambiguities arise when first-party cookies are later used in a third-party context. Consider a user visiting *facebook.com*, which sets first-party cookies for *facebook.com*. On the site *a.com*, which contains an iframe loaded from *facebook.com*, these cookies are now considered third-party. If they are sent back to Facebook depends on the browser: One group of browsers completely blocks third-party cookies, the other group only prohibits third-party websites from *setting* cookies, but does allow them to send *pre-existing* cookies. To which group a browser belongs can change. Roesner et al. [13] report Firefox as the only major browser that also blocks sending third-party cookies.

The most serious issue of blocking third-party cookies is the loss of functionally on most websites that share cookie data with multiple domains [38]. A new approach to provide a similar benefit is to isolate websites from each other, which will be described as **Site Isolation** in this work and is based on our publication [23].

The basic principle of isolation can prohibit some linkage of visited websites and is already used in different varieties:

- Regularly clearing all site data, for example when exiting the browser. Deleting all data from visited websites inside the browser history helps to reduce tracking. Unfortunately, this cannot protect against data leaks involving longterm identifiers like the linkage of an e-mail address or a Facebook account id.

- Using different browsers to isolate some high profile sites from others, for example to isolate online banking from regular web browsing. Site Isolation aims to scale up this approach to *all* visited sites, because it is not feasible to do it manually.

- The private browsing modes that major browsers implemented over the last years provide an additional isolated profile that is only stored in RAM. This protects the private browsing session from the normal browsing session, but has the drawback that trackers can still read all associated cookies in the active private browsing session.

The proposed concept of Site Isolation in this work transfers these principles into the core of the browser to isolate every site by default, similar like using different browsers vendors for each website.

Higher privacy when compared to manual isolation is achieved by combining the strengths of the different isolating varieties and applying them to everyday browsing. Site Isolation principally limits tracking to each individual site and is able to reduce the number of pages tracked by the top 10 most used trackers by over 50%. It cannot provide as much privacy as would be achieved by blocking all third-party cookies, but it breaks less websites. In addition, the isolated browsing cache also limits various cache-based tracking methods and cache-timing attacks, which are not affected by blocking third-party cookies.

In this chapter we will show that with our Site Isolation concept it is possible to isolate the locally stored website state into separate containers and eliminate the ability of trackers to re-identify users across different sites. This is done by isolating HTTP cookies, HTML5 Web Storage, Indexed DB, and the browsing cache. We show that website functionality is less impaired than by blocking all third party cookies, with a comparable level of protection against cookie-based tracking. In addition, Site Isolation reduces the effect of tracking using the remaining storage methods, and secures against CORS, CSRF, and click-jacking attacks, while limiting the impact of cache timing, and rendering engine hijacking.

We implemented an automated Site Isolation system for the Chromium browser to reduce tracking. In order to evaluate our system, we performed a large scale study on Site Isolation effectiveness visiting 1.6 million pages. This was done by crawling the Alexa global top 40.000 websites and clicking up to 50 selected links by our algorithm. Comparing the link selection algorithm to real user browsing behavior data collected by AOL [268], shows a similar page distribution. For the purposes of this evaluation this means our automated data is realistic. We also added a user-defined cookie timeout as well as support for multiple incognito profiles in Chromium.

Moreover, we show that the top suspected trackers all store enough information to reliable identify billions of users and that Site Isolation can reduce the number of pages tracked by 44 %.

5.1 Design Decisions

There exist different options of adding new functionality to browsers, each with its pro and cons. In this section we evaluate various options.

5.1.1 Intercepting Proxy

The least invasive method is not to modify the browser at all and use a proxy server instead to inject code or modify the visited pages otherwise. Examples of systems are *BrowserShield* [280], *Privoxy*[1] and the approach presented by Jang et al. [226]. The benefit of a proxy architecture is the support for all machines in the network regardless of the browser or underlaying system. Disadvantages are the few possible modifications, limiting Site Isolation only in the context of website changes. That would require capturing and rewriting all cookie, Web Storage, and cache access, but no possibility of filtering access to or modifying the browser cache from a regular web page. Further, it is difficult to make this code injection transparent to the websites while not breaking it functionalities. In addition, modification need to consider various browsers, HTML parsers, JavaScript engines and *DOM* implementations.

Even if implementing Site Isolation for HTTP cookies and Web Storage using a proxy, it would come at a great expense and still not isolate the caches.

5.1.2 Extension or Add-On

The most common method providing new features for a browser is by implementing an external *API* (under Chrome: *Extension*, and Firefox: *Add-on*). Both are implemented using JavaScript and have much deeper access to the browser and system functionalities as JavaScript embedded into web pages.

[1] `http://privoxy.org`

The advantages of implementing new features as an extension are flexibility for the user. Installing and removing browser extension can be done easily. Still, because it is not possible to access low-level API's functionalities to manage profile directories for the browser, we could not implement Site Isolation as an extension.

5.1.3 Plugin

Before most browsers used an extension interface, they used an interface designed to support additional file formats embedded in `<object>` or `<embed>` tags.

Netscape Plugin API (NPAPI):

The NPAPI interface is supported by most browsers and allows registration of a plugin for one or more *MIME* types. The browser then automatically tries to find a suitable plugin, when it encounters an unsupported *MIME* type. Technically, NPAPI plugins are dynamic libraries, that expose specific functions called by the browser for setup and destruction. To communicate with the browser, they can call the functions defined by the API[2]. But NPAPI plugins have full control. For example, Adobe Flash 11.2 for Linux (`libflashplayer.so`) is linked against 76 other libraries, including `libnss`, `libgtk`, `libz`, `libXrandr`, `libGL`, `libwayland`, `libxcb`, and `libudev`. This lack of security mechanisms for plugins makes them extremely dangerous. For almost all privacy enhancing features, the MIME type registration architecture makes NPAPI plugins unsuitable to implement our concept.

Pepper Plugin API (PPAPI):

Chromium introduced the PPAPI as an improvement of NPAPI to allow better sandboxing. In contrast to NPAPI, PPAPI plugins run in a sandbox and must use the APIs provided by the browser to access files, perform graphic operations and others. This makes the plugins mostly usable cross-platform [281]. Since it is only supported by Chromium (and Google Chrome), there are not many third-party plugins, other than those shipped directly with Chromium (and some additional proprietary plugins shipped with Google Chrome). Mozilla has expressed no intention of implementing PPAPI[3].

PPAPI keeps the MIME type registration model of NPAPI, so it is also not possible to get code executed on every page. This limits what malicious and questionable plugins could do with the browser, but also makes it impractical to develop a plugin for the purpose of improving privacy on the web.

5.1.4 Modifying the Browser Source

The most efficient and well integrated changes can be made by modifying the browser's source, when using open source browsers such as Firefox and Chromium. We decided to use Chromium, since Chrome has the highest market share of over 60%[4]. With the advantage of having the power of implementing our concept in a popular browser also comes with a few challenges:

- Users need to be willing to switch to our new browser.
- Bug fixes have to be incorporated rapidly to fix vulnerabilities.
- Supplying updates requires the necessary infrastructure.

[2] `https://developer.mozilla.org/en-US/docs/Gecko_Plugin_API_Reference`
[3] `https://wiki.mozilla.org/NPAPI:Pepper`
[4] `http://gs.statcounter.com/browser-market-share`

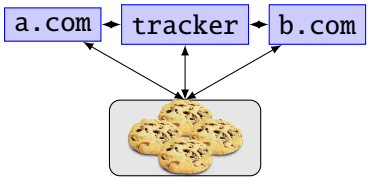

Figure 5.1: Regular storage.

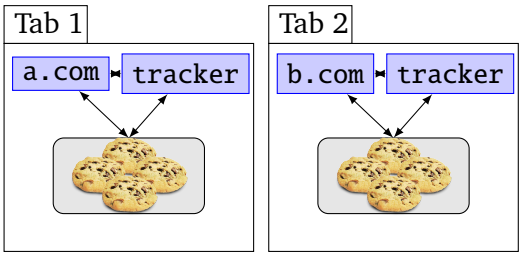

Figure 5.2: Isolated storage.

5.2 Concept and Implementation

Considering the typical methods trackers are employing, like storing an ID on the users computer as well as the privacy impact resulting from re-identifying the ID on multiple sites, we developed various methods to reduce tracking and improve the users' privacy. First, the Site Isolation strategy to isolate tracking elements into container, second, the user-defined cookie timeout and third, the support for multiple incognito profiles.

Site Isolation builds upon the best practices regarding content isolation and pushes the boundaries by applying isolation to every site. Persistent data is stored for each site individually, instead of directly in the browser's profile, together with the data of all other sites visited, see Figure 5.3. With Site Isolation each site gets its own storage partition, which is essentially a miniature version of the regular browser profile, where the data would be stored usually. A storage partition is specific to a certain website, other sites have no possibility of accessing it, contributing to it or even checking if it exists. Site Isolation splits up HTTP cookies, HTML5 Web Storage (formerly Local Storage), IndexDB, and caches. Figures 5.1 and 5.2 illustrate the difference between a global cookie storage and cookies stored in two storage partitions.

The goal of Site Isolation is to avoid automatic data flows between distinct websites. This is typically used to facilitate user tracking, in the form of third-party cookies using JavaScript or when requesting external resources. But it is also used for many mechanisms users expect to work, such as logins or shopping carts.

Mechanism:

For the implementation of Site Isolation we extended the Chromium browser. As an entry point to allow storage partitions for any site we extended the *GetStoragePartitionConfigForSite* method of *ChromeContentBrowserClient*. By default, this function checks if the URL has the *extension://* scheme and if isolation is enabled for the extension. It was extended to enable all sites to be isolated, see Listing 5.1. The storage partition class itself was extended to contain a label, which can be shown to the user, to identify the current storage partition. This value corresponds to the host name and is the same as the subfolder used for the storage partition, as seen on disk (illustrated in Figure 5.3).

User Interface:

We implemented the user interface by extending the easily accessible site preference dialog of Chromium. The UI opens by clicking the page icon in the address bar. The existing dialog is used for example to disable cookies, JavaScript or other elements for each individual site. As shown in Figure 5.4 we added a new entry *Force Isolation*, under the *Permissions* sections. The additional entry allows the user to set if a website should be isolated or not. This makes the isolation feature very discoverable, because users already using this dialog to black- or whitelist access to cookies or plugins manually and are already familiar with it. In addition to enable or disabling isolation, the currently active storage partition is shown in the UI, to see where the data on the current web site will be stored. Since the dialog is platform-specific, it was implemented for the GTK back-end (first), so the additional entries are only in the Linux and Chromium OS version.

```
void ChromeContentBrowserClient::GetStoragePartitionConfigForSite(
 content::BrowserContext* browser_context,
 const GURL& site,
 bool can_be_default,
 std::string* partition_domain,
 std::string* partition_name,
 bool* in_memory) {

 // [...] Initialization
 // [...] Set up storage partitions for packaged apps and extensions

 // Allow *all* sites to become isolated
 if (can_be_default && site.has_host()) {
  Profile* profile = Profile::FromBrowserContext(browser_context);
  if (shouldBeIsolated(profile, site))
    *partition_domain = site.host();
  }
  // [...] Assertions
}
```

Listing 5.1: Modified `GetStoragePartitionConfigForSite` method to give each site a distinct storage partition. The policy decision of which sites to isolate is delegated to the `shouldBeIsolated` method.

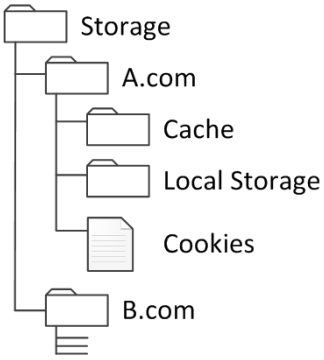

Figure 5.3: Folder structure.

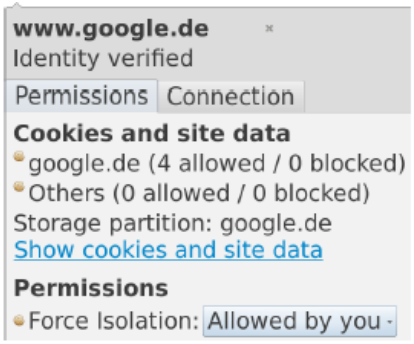

Figure 5.4: User Interface.

User-Defined Cookie Timeout:

In order to provide better user privacy, a user-defined cookie timeout was implemented also in the site permission dialog - see Figure 5.6. Instead of removing cookies after the corresponding tab is closed, cookies will be deleted if they have not been read or changed for a user-defined period of time. Removing cookies automatically from tabs that are still open can result in a better protection, for example when the tabs have not been used for a long time. By giving users the option to manually limit the lifetime of cookies, which previously have been saved by the third-party for years, can now be edited or deleted automatically. The per-site timeout can also be set to infinite, in order to white list important sites. A downside to this strategy is that trackers appearing on many sites, will still have a big privacy impact and the least affected by this change. If third-party cookies are constantly used on websites, they will not be deleted unless the user stops browsing for a while.

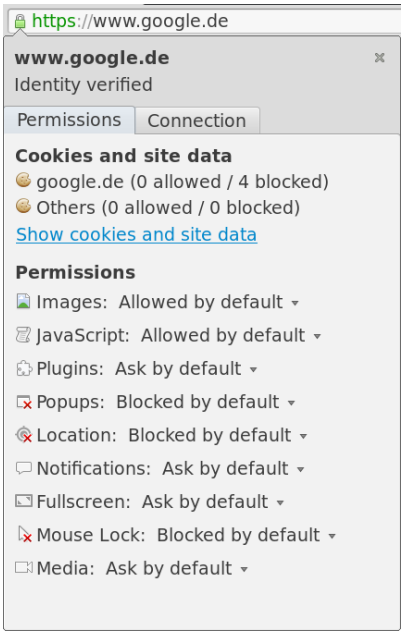

Figure 5.5: Original preference dialog.

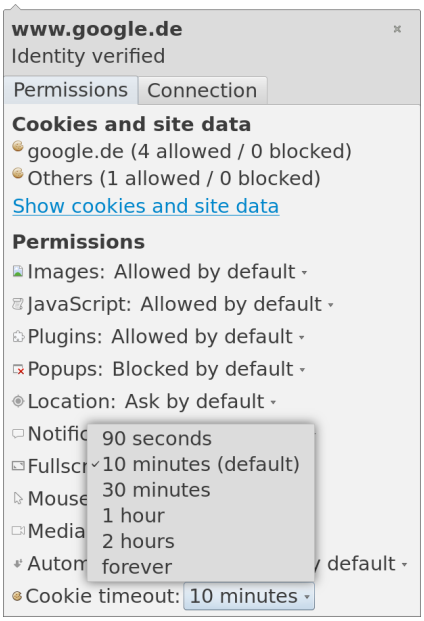

Figure 5.6: Cookie timeout feature.

```
bool shouldBeIsolated(Profile* profile, const GURL& url) {
HostContentSettingsMap* map = profile->GetHostContentSettingsMap();
ContentSetting setting = map->GetContentSetting(
url, url, CONTENT_SETTINGS_TYPE_ISOLATION, std::string());
return setting == CONTENT_SETTING_ALLOW;
}
```

Listing 5.2: Implementation of `shouldBeIsolated`. It uses the `HostContentSettingsMap` to decide if a site should be isolated or not. It looks up the saved data by retrieving the value of the newly added `CONTENT_SETTINGS_TYPE_ISOLATION`.

5.2.1 Storage Policy

With the isolation mechanism in place, a policy is required to decide which tab should map to which storage partition. Using the *host name* of the visited site as the storage partition name turned out not to be ideal (using Chromium's *GURL::host()* method), because it often results in bigger storage partitions than necessary. For example, *a.blogspot.com* and *b.blogspot.com* are both put together into *blogspot.com*, although they are *owned* by different persons. This is similar to some registrars allowing or forcing the registration of domains not directly under the Top- Level Domain (TLD), such as *bbc.co.uk*. This could allow dangerously far-reaching cookies to be set.

The solution is Mozilla's public suffix list[5], which is already integrated in Chromium. It can be used to find out the *top private domain* (the name used by Google's Guava library[6]), which is the largest part of the domain, where cookies are allowed to be set to. For example, *a.b.c.co.uk* has the top private domain *c.co.uk*.

In our policy the user can also enable/disable the isolation for specific sites. Allowing more customization, such as to switch to arbitrary storage partitions, is possible and more powerful, but also difficult to make usable. Another improvement that may have merit would be to allow making the storage partitions finer or coarser, i.e., changing how many subdomains are used to construct the storage partition

[5] `http://publicsuffix.org`
[6] `https://code.google.com/p/guava-libraries/wiki/InternetDomainNameExplained`

name, like to allow to isolate *www.google.com* from *maps.google.com* or *play.google.com*. To implement this policy, the mechanism for the existing per-site preferences was reused. This results in a very compact definition of the *shouldBeIsolated* method, shown in Listing 5.2.

5.2.2 Support of Complex Interaction

Considering realistic browsing sessions where users click links to other pages or where websites communicate with other domains, we have to make sure not to break features provided by websites, as soon as there are multiple storage partitions involved. In the next paragraphs we show how Site Isolation reacts to different browsing scenarios to make the site (or feature) usable.

Cross Subdomain Cookies:
Providers often use different subdomains for their services like POST the login data of *mail.a.com* to *accounts.a.com* and get a session ID as a first party cookie. This does not involve third-party cookies or multiple storage partitions, so it cannot break our concept. The proposed enhancement to allow the user to make the isolation finer-grained, however, would (when enabled) put *mail.a.com* and *accounts.a.com* in a different storage partition, which will most likely break the login. Not switching storage partitions on redirects fixes this problem, by keeping both cookies in the same storage partition. In that case, using *docs.a.com* would require a fresh login, as expected, considering the user has explicitly opted to have all subdomains of *a.com* isolated.

Unusual Logins:
Google uses *accounts.google.com* to validate the data entered into the login form on *mail.google.com*. Since only the subdomain is different, it is possible by setting the cookie for *.google.co*m from *accounts.google.co*m, that will be sent in subsequent requests to *mail.google.com*. Using the login also for YouTube, which will not be sent because of the same origin policy, Google uses a workaround. Here, *accounts.google.com* sets its own cookies and then redirects to a URI on *youtube.com*. Then, the server sets the same cookies for *youtube.com* and redirects back to *mail.google.com*. These redirects are hardly noticeable, because they are done server side using HTTP 302 Found and the Location header. Switching the storage partition during the redirects would set the login cookie in the *youtube.com* storage partition, keeping it would set it in the *google.com* storage partition. Both options are viable and have their own advantages, which will be discussed in Section 5.2.3.

Social Plugins:
The variety of social media plugins like Facebook *like* or Google *+1* buttons are an important use case for Site Isolation. Considering a *like* button on a website the user has to perform a login at some point to use the service when clicking the button.

If this login is implemented as an overlay on the main site, which uses further iframe/AJAX techniques to log in without redirecting to a different site, this will work fine with Site Isolation. The address bar and therefore the storage partition was never changed, resulting in the user being logged in, but only for the actual site. If implemented as an iframe, as on most pages, the iframe runs in the same storage partition as the main page, but neither the plugin has access to the page it is embedded on nor has the page access to the plugin iframe's content. When clicking, a new page is opened, avoiding trivial login phishing. This works also with Site Isolation, if the social plugin is implemented robustly. The following possibilities will cover most existing interaction strategies.

Simple Link:
The social plugin always uses a link directly to the intended destination, regardless of any cookies set, for example linking to */like?page=a.com*. This works, because when clicking the link, the storage partition switches and the user is logged in (or will be prompted to login on the new page). Note, that

switching or not the storage partition does not have any effect on redirects here, since there are no (cross-domain) redirects involved.

Login Detection:
The social media plugin detects that there is no login cookie in the currently active storage partition and behaves differently. It might create a link with enough information to redirect back to the original destination, e.g. instead of linking to /like?page=a.com, the link now points to */login?redirect=%2Flike%3Fpage%3Da.com*. The login page will be loaded with a different storage partition and can automatically redirect back to */like?page=a.com*. To avoid redirect problems, navigation events from inside frames could keep the currently active storage partition if the user wants to.

5.2.3 Storage Strategies

To avoid some of the potential website functionality issues, there are many enhancements possible, that can be changed in an advanced configuration mode. But it is important to keep their side effects in mind, especially on usability. In the following we explain the configuration settings behind the default storage strategies.

Keep Storage Partition on Automatic Redirects:
In the current implementation we only change the storage partition in response to user-initiated actions. This seems to be counter-intuitive, as generally more storage partitions lead to more isolation and privacy. However, this solution will isolate an automatic redirect, which might store third-party tracking cookies, even when they are disabled by the user. Keeping automatic redirects in the same partition will improve the functionality of websites and is more user friendly, though users will have to login multiple times. Otherwise the user might get confused about already being logged-in on another site despite using isolation.

As explained in Section 5.2.2, Google for example uses redirects across different domains to set first-party cookies on multiple domains, which will all stay inside one storage partition. Here the user benefits from keeping the intended login into multiple services such as Youtube and GoogleMail, even it might confuse the user being logged into another service while still using Site Isolation. If we would have chosen different, it would make the web application not usable. Another drawback of keeping the storage partition on automatic redirects is that it does not limit tracking during the redirect process. Luckily, this does not effect limiting tracking on iFrames, which will put the cookies from the iFrame domain into another container then the original website.

Challenges arise when elements are styled like links but are used in combination with a `onclick`-handlers, to make them look and behave like links. While this is technically a JavaScript initiated redirect, it should be considered as user-initiated.

Keep Storage Partition when Clicking Links Inside Frames:
A different strategy is to ensure that pages linked from inside frames have access to the same set of cookies as the frame did. In other words, it means keeping the storage partition, if the redirect originated from inside a frame. It has similar usability problems as all stateful storage partition selection has, namely that visiting exactly the same URL can result in a different storage partition being used. But it avoids the issues of a frame showing a user to be logged in, but after clicking on a link the user is suddenly logged out. Conversely, it also avoids a frame showing a user to be logged out, then switching to logged in after clicking on a link. This might be a usability hindrance, since previously there was no additional login required. But it is also easy to construct an example where it is beneficial. Usually, it takes just a single misplaced click to *like* a page on a dubious website accidentally. By keeping the storage partition when clicking links inside frames, users are not logged in on the first try to *like* a page. They can now decide to go back or log in, in which case future *likes* from the same page will work directly.

Manual Storage Partition Selection:

In the case where automatic selection should not work, users can take control of how storage partitions are assigned. In the advanced user mode, the complete storage partition name can be specified on a per-site basis.

Whitelisting to allow Communication Between Sites:

As an alternative the user can define which websites should share a storage partition by selecting it in a list. The list is built from the set of domains referenced on the current page as resources and the domains it has links to. Here, the user can select sites to be part of the same storage partition.

Additional Link open Modes:

Additional options to open links that explicitly keep or change the current storage partition can help power users to manually work around defects or improve isolation further. These would be available in the context menu of links, as an additional entry, besides opening the link in a new tab, new window, or incognito window.

Keep Storage Partition Mode:

Instead of using one of the mentioned modes, a simple switch to disable switching between storage partitions temporarily can help fixing website isolation problems. Activating the *storage partition lock* and reloading the page should fix problems caused by Site Isolation.

Multiple Incognito Profiles:

The private browsing features that are integrated into most of the browsers, essentially provide a convenient isolation for two profiles. Both profile types are isolated from each other, meaning that websites opened in the incognito profile cannot read cookies or local storage from the base profile and vice versa. Two sites opened in two incognito tabs, however, can freely share data. Unlike Site Isolation, it requires explicit manual intervention to activate and is less powerful. In Site Isolation we combined both strategies and enabled for each incognito window its own profile. Closing an incognito window would always cause the associated data to be deleted, where previously the data would only be deleted if it was the last incognito window.

Summary:

The built-in storage partition mechanism of Chromium 29 was extended with different storages strategies to isolate Cookies, Web Storage, IndexDB and caches of each visited site automatically, based on its top private domain. The benefits from these changes are:

- Improved privacy by limiting cookie-based web tracking to each visited site. The effectiveness of the privacy benefits will be shown in Section 5.3.1.

- Improved security by isolating websites by default, making it harder for malicious sites to affect security-critical sites opened in the same browser. CORS, CSRF, and click-jacking attacks are no longer possible.

- Improved cookie management, while the locally stored data is split up per site and can be cleaned up selectively.

- Protection against stored or cached tracking mechanisms like Web Storage, IndexDB, ETags or cached images [12], [11].

5.2.4 Isolation Classifications

Site Isolation operates on different boundaries than previous isolation strategies. Compared to other isolation strategies Site Isolation has numerous benefits and advantages.

- Compared to *private browsing*, Site Isolation works for every site and is not limited to two partitions. In private browsing, trackers can still retrieve the data from different sites opened in a private browsing session [157]. Site Isolation prohibits this behavior by using a new storage partition (on disk) for each visited website.
- In contrast to using *multiple browsers*, Site Isolation works automatically, so the user cannot forget to use it. Multiple browsers can be used to have different states for the same site, but this can be achieved more easily by using multiple incognito profiles, presented in Section 5.2.3.
- Site Isolation works orthogonal in respect to clearing site data (cookies). It partitions in space, while clearing cookies partitions in time. Both can be used complementary to improve privacy further.

Benefits compared with third-party cookie blocking:
Site Isolation prevents trackers from using a single identifier for users across their entire browsing session. Each time a new site is visited, the cookies will be read from a different, initially empty storage partition. Trackers can no longer follow the user around the web, because the user appears to be a different one on each site. As a result, Site Isolation has largely the same effect as blocking third-party cookies, despite being conceptually different. This results in the following advantages:

- Third-parties can easily check if third-party cookie blocking is enabled and try to use a different tracking method. Site Isolation behaves in this example as if third-party cookie blocking is disabled, and an isolated cookie will be sent.
- Blocking third-party cookies can easily break websites [38]. By not blocking cookies, Site Isolation helps to retain the website functionalities.
- Site Isolation can interfere with multiple websites in a similar manner, but only if they actively redirect the top level window. Only when the domain (host) in the address bar changes, there are observable effects. Redirections and links to the same domain and cross-site frames will always keep the storage partition, meaning that no breakage can occur. Facebook Apps, for example, are always running in a third-party iframe on *facebook.com*. If they are using cookies, blocking third-party cookies will break them; with Site Isolation, they continue to work.

Positive effects of cache isolation:
The isolation of the browsing cache severely limits the scope of cache-timing attacks [159, 282] that can be used to get information about the browsing history. However, it does not impact other kinds of timing attacks to steal history data, such as the recently discovered rendering time attack [75]. Isolating caches can also limit cache-based tracking methods where trackers try to store a random identifier in the browser cache and retrieve it from the cache later.

While Jackson et al. [266] propose a same-origin restriction for cache access forbidding third-party content to read from the cache, Site Isolation already eliminates most of the privacy issues. Because every site has its own cache, tracking is limited to each individual site. Just as with cookie-based tracking, trackers cannot re-identify a user on a different site by using cached data.

Summary:
Site Isolation provides a similar privacy benefit as blocking third-party cookies, but it is more practical, as it breaks less websites and enables new privacy features like user-defined cookie timeout and multiple

incognito windows. It can limit different tracking strategies, such as cache-based or using additional storage options besides cookies, for example Web Storage, IndexDB or tracking images. Additional security is provided by avoiding CORS, click-jacking, CSRF and limiting history hijacking, cache timing, and rendering engine hijacking, as already analyzed by Chen et al. [283].

5.3 Evaluation

Demonstrating the performance of Site Isolation we conduct an analysis of the reliability, the effectiveness, and correct handling of real browsing behavior. To test the reliability of usual browsing we use a script to open 1.6 million pages automatically in the patched Chromium. The next aspect is effectiveness. Here, we compare this data in normal browsing mode and with Site Isolation. Lastly, it is also important to ensure that Site Isolation does not break any widely used web techniques that provide features to users, while still hindering web tracking. Automating such tests is hardly possible and depends on a solid definition of *break*, which is why general browsing examples were evaluated manually.

Test Environment:
Chromium was compiled and executed on a Linux (Ubuntu 12.04 LTS) system. A Java-based control program started and managed multiple Chromium instances. The test machine, powered by an Intel Core i7, 16 GiB RAM and a 50 MBit/s Internet connection, handled 40 parallel browsers without problems and 75% CPU usage.

Automation using a Control Program:
A Java-based control tool starts and automates the Chromium instances. These are directed to load a custom extension, which sends the following data back and receives new commands:

- The list of all HTTP requests made by the current page, which are used later to identify the biggest privacy leaks, and the content of the address bar are sent whenever it changes. This is done to be able to handle redirects, which could lead to a different site, such as redirects between http and https, subdomains or to forward to the corresponding country TLD.

- The list of all links on the loaded web page. To consider links that are added by other scripts while waiting for the page to be completely loaded, a configurable delay is introduced, before gathering links. We also filtered and validated this list to discard links like "*#top*" or invalid targets.

- The control tool replies with a random URI from this list. The extension now sends a click event to the associated DOM element, so onclick actions do not get skipped and the correct HTTP Referer is sent with the request.

- Navigation errors and timeouts are directly reported by Chromium to the control tool, and a random previously visited URI is opened.

After each browsing session, Chromium's internal databases in the profile directory are read out. Both Cookies and HTML5 Local Storage are stored in a SQLite databases. The cookies are stored in the profile directory in a file named `Cookies`, Local Storage is saved per-site in the `Local Storage` directory. Adobe Flash was installed and enabled for the tests, and the *Pepper Plugin API* (PPAPI) version distributed with Google Chrome was used. The *PPAPI* version of Flash stores *Local Shared Objects* in the `Pepper Data/Shockwave Flash/` directory of the browser profile, so it was copied over from Google Chrome 29. It involved copying the `libpepflashplayer.so` file and pointing Chromium to it using the `-ppapi-flash-path` flag.

The corresponding data inside the storage partitions is read, too. They contain their own `Cookies` and `Local Storage/` databases, located in the `Storage/ext/` subfolder. Unfortunately, the PPAPI Flash

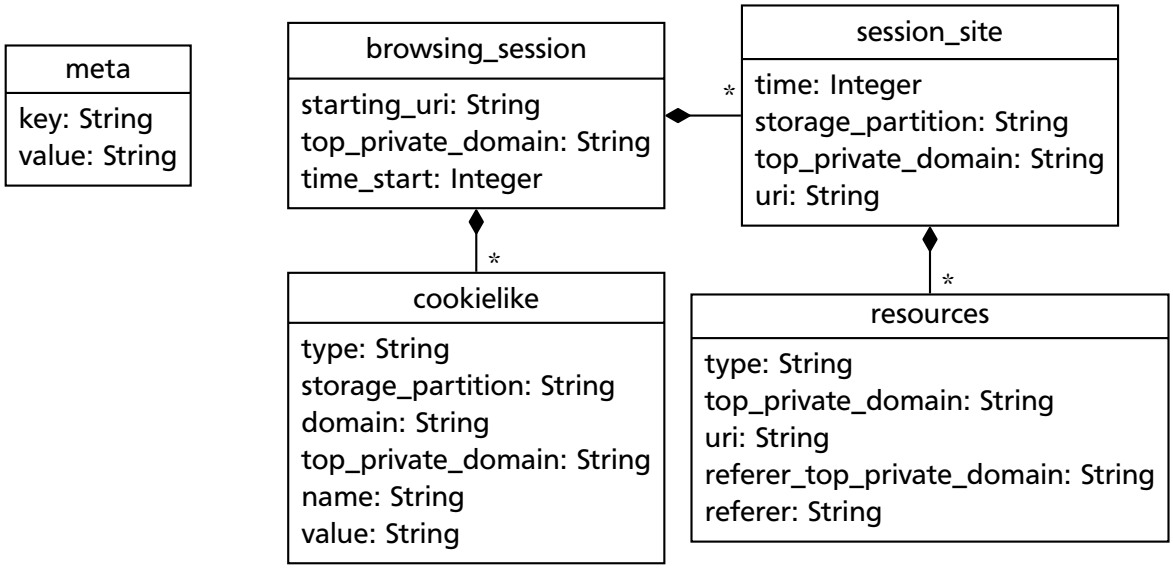

Figure 5.7: Database structure of the crawling results in form a UML class diagram.

plugin doesn't use storage partitions and always saves the LSOs directly in the profile. It seems fixable, however, since the PPAPI version is sandboxed and has to access the disk via Chromium.

The database structure is visualized in Figure 5.7. The *meta* table stores all configurable settings used for the crawling and the start and finish time. Each browsing session, i.e., each Chromium instance started, is represented by an entry in the *browsing_session* table. Each cookie and other cookie-like objects are stored in *cookielike*. They are only associated to the browsing session, because they are read out after the browser exists and not on every page view. Each page that is visited by each browser is listed in the *session_site* table. Each resource loaded by browser when visiting a page is logged in *resources*.

5.3.1 Browsing Sessions

Automated browsing sessions are used because they are much more feasible and comparable than using browsing sessions by real users. There is also an inherent privacy problem of using users' generated browsing histories. Simply recording all visited URIs is not enough, because many websites' URIs are valid only for a short period of time. After a few days or even earlier, the content of these pages changes, typically to an error page that links back to the home page. To get around this problem, the content would have to be recorded in addition to the URIs, which creates further privacy and technical issues. To avoid these issues, the main evaluation uses automated browsing sessions. These are, just as recorded user sessions, not perfectly reproducible, but can produce a vast amount of data in a short time, without much manual effort.

Biased Crawling

As entry points for the browsing sessions the Alexa global top 40.000 list is used. Each domain is then selected as the entry point of a browsing session. After opening the home page of each domain, the work flow described above applies, and selected links are clicked until 50 pages have been visited. At this point Chromium is closed, the data of the profile read out and added to the database. When finished, another session is started using an empty profile, until all domains have been visited.

In order to avoid problems with completely random link selected crawling and to produce data that is more useful the following mechanism was used:

Domain	Present on pages (%)
google-analytics.com	(54.0 %)
doubleclick.net	(26.4 %)
facebook.com	(23.2 %)
google.com	(21.6 %)
gstatic.com	(16.4 %)
fbcdn.net	(14.2 %)
twitter.com	(12.1 %)
googlesyndication.com	(11.2 %)
scorecardresearch.com	(11.1 %)
googleapis.com	(7.8 %)

Table 5.1: Top 10 domains referenced most from other pages.

Domain	Sessions with Cookies
doubleclick.net	27,844 (97.8 %)
google.com	22,951 (75.6 %)
twitter.com	20,828 (92.2 %)
scorecardresearch.com	19,263 (98.7 %)
youtube.com	15,339 (96.9 %)
adnxs.com	13,417 (98.9 %)
quantserve.com	11,495 (97.3 %)
yahoo.com	10,951 (97.1 %)
addthis.com	10,377 (90.8 %)
yieldmanager.com	9,569 (84.8 %)

Table 5.2: Top 10 domains that had the most cookies set in browsing sessions.

1. Only pages that have not yet been visited in any other browsing session are considered for link clicks.

2. Domains that have been visited less often than others are favored, instead of selecting them completely at random.

3. To avoid visiting a lot of obscure domains that users are not very likely to ever visit, domains ranking high in the Alexa list are favored.

The resulting list has the links to the least visited sites first, but unpopular sites are slightly disadvantaged. The goal of this mechanism was to visit more of the smaller sites at the expense of larger ones. As pre-tests have shown, completely random link selection resulted in visiting Twitter, Google and Facebook links in 30% of all crawled pages (since those domains occur the most on websites), but is unlikely to result in realistic user behavior and therefore no useful data.

We implemented the biased link selection by choosing the index for the list as $i = \lfloor r^b \cdot n \rfloor$, with r being a random floating point number in the interval [0.0, 1.0[, b being the bias exponent, and n being the number of elements in the list. The value $b = 1.2$ is used to provide just a slight bias towards the first elements of the list. Note that the sheer number of links to a site still has a big impact on link selection.

In result of the biased link selection, the cumulative distribution of the domains visited is plotted in Figure 5.8 as shaded green area (bordered with green crosses). When compared to a real world page view, data by AOL users reformatted from [268], shown as rounded blue circles in the same Figure 5.8, a similar shape can be noticed (just with a less distinct curvature). For the purposes of this evaluation, this means the biased link selection is somewhat realistic compared to real world page views.

The curvature in the range from 0 to 50 page views can be explained by the 50 link click limit and corresponds to the end of the 40,000 sites, that received very few incoming links from other sites (or that were left early in the session because an external link was followed).

Crawling Results

To evaluate how Site Isolation performs, a few metrics are needed. Every request to an external domain is considered a tracker, except user initiated clicks, because they direct to a different storage partition. The more domains are referenced by websites, the bigger their tracking potential. Table 5.1 shows the results of the 10 domains with the biggest tracking potential (out of 1,608,038 visited pages).

Tracking Data:

The occurrence of external content and potential tracking by setting cookies is not a sufficient definition of tracking. To actually perform tracking, the trackers have to be able to re-identify users. They can either use static user data directly, such as the IP address or browser fingerprinting, or they have to store a custom identifier and retrieve it later.

In this section, the latter method is investigated. Table 5.2 shows the top 10 cookie setting domains. When comparing with Table 5.1, the Content Delivery Network (CDN) domains are notably absent, because they do not set any cookies. Also note that *google-analytics.com*, which has by far the biggest tracking potential, in fact does not even set a single tracking cookie for its domain. Instead, it uses first party cookies on each website it is embedded. To find out which of these cookies are used (or are usable) for tracking, we calculate the information content of cookies.

In theory, just 33 bits are enough to uniquely identify 8.5 billion people. To calculate the information content of a web site's cookies, Shannon's information entropy [284] was used. The entropy of the cookie data is equivalent to the number of bits required to encode one symbol (i.e., a byte) of the cookie data and can be calculated using $H = -\sum p_i \log_2 p_i$. Multiplying the entropy H by the total number of symbols results in the size that the data can be compressed to when using Huffman coding [285]. In order to avoid over-interpreting cookies with a high information content, like web sites storing settings with the similar data for all users, the data gathered from multiple sessions can be used to differentiate between random and highly structured cookies. This can be performed by calculating the information content of the concatenation of a web site's complete set of cookies in all sessions.

The actual string that will be analyzed is a concatenation of the sorted cookie names followed by the sorted cookie values (skipping duplicates). The benefits for this method are:

- All cookies are concatenated because information might be split across multiple cookies.
- Names are included because information may only reside in the cookie names.
- Identical names and values are included only once, to get a smaller size estimation from Huffman coding, which does not check for identical blocks in the uncompressed data.
- Almost identical names and values will not compress well when just using Huffman coding. To avoid this problem a state of the art compression algorithm is used in addition.
- To improve compression, the names and values are sorted to get similar names/values next to each other.

The additional compression algorithm used is LZMA2 [286], because of its great compression ratio [287] and because it is based in a different principle than Huffman-coding. In the calculation LZMA2 tends to win over Huffman-Coding in 39% of the cases, typically when data is large enough to offset the overhead, that is ignored for Huffman-coding. The lower estimate of both algorithms is then used as an estimation of the information content, and if a suspected tracker domain has a very low content, it is analyzed manually. Table 5.3 summarizes the findings, showing that all of the heavily referenced and cookie setting domains can use their cookies to track users (high information bit ratio).

Other researchers [28] confirmed our findings, showing that 80% of tracking cookies use more than 35 characters in the information value field. In addition, they show that 90% tracking cookies have an expiration date greater than one day.

Isolation Effectiveness:

Site Isolation impacts the previously gathered data in a number of ways, by not using the global profile to store data but instead storing it on a per-site basis. Each of the 40,000 cookie stores (sessions) can now have an unlimited amount of storage partitions. The total number of distinct storage partitions seen across all sessions was 94,701. But since not every storage partition is present in every session, there were only 176,330 non-empty storage partitions in total. The main effect of Site Isolation is that

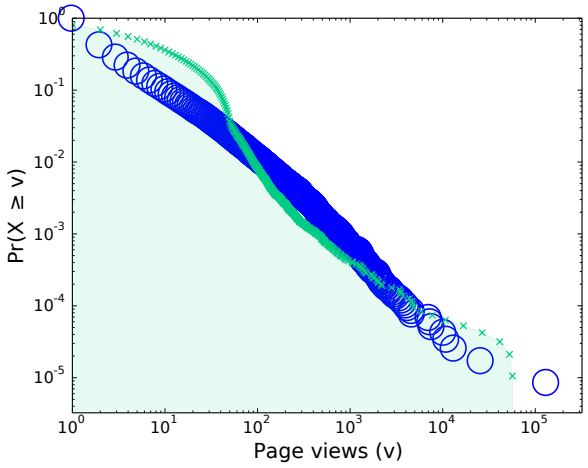

Figure 5.8: Pages crawled (crosses) / viewed by AOL users (blue circles).

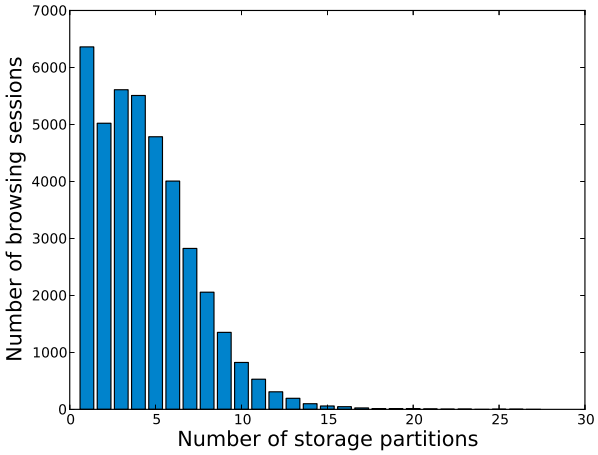

Figure 5.9: Storage partition usage in browsing sessions.

Domain	Cookies in Sessions		Pages tracked	Cookies in S. Partitions		Pages tracked*	Inf. bits
doubleclick.net	27,844	(97.8 %)	414,808	30,104	(51.2 %)	217,320	69
google.com	22,951	(75.6 %)	263,019	24,742	(35.4 %)	123,170	375
twitter.com	20,828	(92.2 %)	178,853	22,624	(49.6 %)	96,178	143
scorecardresearch.com	19,263	(98.7 %)	175,727	20,576	(62.4 %)	111,169	35
facebook.com	8,974	(32.0 %)	119,628	9,390	(15.2 %)	56,743	97
quantserve.com	11,495	(97.3 %)	99,217	12,280	(66.8 %)	68,096	54
adnxs.com	13,417	(98.9 %)	93,785	14,117	(72.9 %)	69,167	357
addthis.com	10,377	(90.8 %)	80,840	10,890	(64.3 %)	57,289	153
youtube.com	15,339	(96.9 %)	56,598	16,012	(66.7 %)	38,929	272
baidu.com	2,152	(95.3 %)	47,923	2,606	(45.6 %)	22,939	104

Table 5.3: Effects of storage partitions on the top trackers. The pages tracked metric dropped by 44 %, compared with not using Site Isolation.

tracking cookies are not global to the browsing session and are instead local to each storage partition. As the evaluation of cookies show, trackers assign multiple different identities per session, making the tracking effectively limited to a single site, no more than what would be possible using completely passive server logs or using first-party cookies. The number of storage partitions per browsing session is visualized in Figure 5.9. The statistics indicate that on average the storage partition is switched every 11.2 pages. As each switch means all trackers up to this point will continue with an empty set of data, this sets the limit for how long users can be tracked. An interesting and unexpected result is the high percentage of empty storage partitions. These numbers, shown in Table 5.3 were generated by calculating the cookie statistics based not on the whole browsing sessions, but based on the individual storage partitions.

The *Cookies in Sessions* metric (from Table 5.2) was supplemented by *Cookies in Storage Partitions* and the *Pages tracked* metric was recalculated. While the absolute number of used cookie stores is always higher in the storage partition case, the total number of cookie stores where the domains were referenced is much higher still. This results in a comparatively huge number of storage partitions that do not have any cookies, while many trackers previously had cookies in nearly 100% of sessions. This has a corresponding drastic effect on the pages' tracked metric that fell immensely, indicating that Site Isolation can completely avoid various tracking mechanisms and is not merely reducing it to within-site tracking. All with fully functioning websites.

5.3.2 Discussion

Using Site Isolation brings a slightly higher resource usage on disk space, CPU, RAM and bandwidth. As already analyzed by Chen et al. [283], showing that visiting 12 sites in isolated storage partitions instead of a single profile increased the disk usage 4.5-fold, while RAM usage grew by just 1 MB. As our experiments confirm, no noticeable performance loss could be noticed while using Site Isolation on the test computer.

Currently the storage partition name is determined by the address bar URI. One option that could improve compatibility and isolation, is to make the selection stateful and consider how the browser arrived at the current page. If a normal link was clicked, the storage partition should change according to the address bar, as usual. If a HTTP, HTML or JavaScript based redirect was used, however, keeping the last storage partition can be beneficial. Reducing the number of storage partition switches cannot make compatibility worse, but tends to reduce isolation, as no switches also mean no isolation. But these redirects are often used to set cookies, giving websites an easy way of setting identical cookies in multiple storage partitions. Another possibility is to use more information for selecting the storage partition name, for example by using numerical ids and an associated database. Multiple sites could then be mapped into a single storage partition, to allow a user-configurable whitelist. Any changes that affect the behavior of the storage partitions itself, for example allowing sites to read from multiple storage partitions at the same time, are not considered.

Flash usage is widespread in the current web but dropping in recent years. Unfortunately Chromium does not yet confine the Flash plugin to storage partitions yet, which makes data exchange across storage partitions possible using Flash LSOs. However, instances of Flash LSO based tracking are rare, and in our study only 1,886 domains (0.11%) in total used Flash LSOs. Comparing the top domains using Flash LSO to the tables of sites present on most pages, or the domains having cookies in most sessions, no overlap domains (in the top 10) can be found. We are not aware of any conceptual issues with integrating the PPAPI[7] Flash plugin into the Site Isolation concept, but actually implementing it was left as future work.

5.4 Related Work

When comparing Site Isolation to tracking prevention tools such as Ghostery[8] or AdblockPlus[9], various disadvantages in those tools can be identified. One drawback is that they rely on detecting tracking elements and reactively blocking them. Therefore, elements not categorized as trackers will not be blocked and can still be used for tracking. In addition, if legitimate elements are falsely classified (false positives) as trackers and blocked, website functionalities may break. The most serious privacy problems exist in the commercial purpose of such tools, i.e., a third-party can avoid that certain tracking elements will be blocked (by making a contract) [61], or by the utilization of the gathered user data [69] to allow building a user web profile. With Site Isolation elements from a website are isolated from other websites and will stop cross-domain tracking without breaking the website functionalities. Furthermore, website owner can use ads to earn money without enabling trackers to re-identify the user in order to learn the user's web behavior.

The *doublekey* cookie concept proposed on Mozilla Bugtracker [288] is similar to Site Isolation. Its core idea is to replace the cookie domain with a tuple consisting of the cookie domain and the domain of the currently open web page. As a result, each visited first-party site has its own storage location for third-party cookies. In 2014 the TOR project was in the process of implementing this concept [289], finishing it in 2018. Site Isolation is a superset of this concept, not only isolating cookies, but also other storage methods. Thereby, our solution is limiting many more tracking avenues such as cache based tracking (e.g. using ETags or modified PNG images), Web Storage and others [11]. Site Isolation is also

[7] https://code.google.com/p/ppapi

[8] `https://www.ghostery.com`

[9] `https://adblockplus.org`

extensible for storage methods not yet isolated, such as data from Flash. Since no other implementation has been done before, we were able to present the first large scale study on this topic and analyze the effectiveness of this concept.

Chen et al. [283] investigated which security benefits are possible using multiple, separate browsers and showed how they can be brought to a single browser using *App Isolation*. Here, the researchers implemented entry-point and state isolation for Chromium, which sets a whitelist of allowed entry-points into a website, to avoid reflected XSS, session fixation, Cross-Origin Resource Import and CSRF attacks. The disadvantage is that this whitelisting has to be done manually for each website and every entry point separately. In comparison, our State Isolation brings the same security features and confines each website automatically into a separate, from other websites isolated data container, where the cache, cookies and other locally persisted state are saved. On a high level, App Isolation is targeted at protecting users from manipulation of a few select high-profile sites, while Site Isolation is targeted at protecting users against privacy breaches resulting from third-party trackers embedded on all pages with no changes on the web.

Other strategies to gain security by isolating web elements like plug-ins or JavaScript and better control the execution of those exist [290, 291, 266, 292]. Unfortunately, these strategies do not consider privacy aspects related to tracking, since they allow the communication between storage elements like cookies, which still have access to all tracking related elements. In contrast, our isolation strategy benefits from the same security aspects while protecting the users privacy from being tracked online.

5.5 Summary

As seen in the evaluation, Site Isolation has a huge effect on the abilities of web trackers to follow users across the web. It limits their ability to re-identify users switching to a different site, making tracking much less invasive. On average, the storage partition was switched every 11.2 pages, providing credibility to that claim. On top of that, the number of pages that can still be tracked, was shown to be reduced by 44%, compared the number of pages tracked by the 10 most prevalent trackers when not using Site Isolation. This was unexpected and suggests that external trackers do not set tracking cookies on all occasions, which helps Site Isolation achieve this huge reduction in the pages' tracked metric. The findings confirm the tracking statistics of Roesner et al. [13], that the biggest trackers are now present on over 20% of web pages. As research shows [293], Facebook has stopped indiscriminately tracking even non-users. It set cookies in just 32.0% of sessions it was referenced, dropping further to having cookies in only 15.2% of storage partitions.

The evaluation of countermeasures against Site Isolation in Section 5.3.2 demonstrates that the currently known evasion techniques are impractical. Neither using Flash LSOs nor redirects to set first party cookies are well suited for avoiding Site Isolation on a massive scale. Finally, compared with blocking third-party cookies, Site Isolation is more user friendly and does not depend on the definition of what exactly constitutes a third-party cookie. Combined with the security benefits of preventing CORS, CSRF, click-jacking, and limiting the effects of cache-based tracking, cache-timing attacks as well as rendering engine hijacking, Site Isolation is a promising approach for improving user privacy and security.

6 Robust Browser Fingerprinting Protection

Browser fingerprinting is a technique for uniquely identifying a user through slight variations in the system setup, which are retrievable by websites mostly using JavaScript, Java or Flash scripts [51]. The differences originate from various hardware, network, or software configurations that a user's computer may have [45, 49, 52, 53, 47]. These *fingerprinting features* are widely used to track user behavior across different websites [30, 54] and to establish detailed user profiles [24, 35] for targeted advertisements [32].

Mayer [56] and Eckersley [46] demonstrated that it was possible to uniquely identify 94% of 470,161 users by rather simple fingerprinting techniques. Further studies [57, 50] revealed the prevalence of browser fingerprinting in real world applications and showed that pretty sophisticated fingerprinting methods exist. Fingerprinters commonly use detailed information, such as system fonts, plug-in version numbers, or even differences in graphic rendering engines (using the HTML5 canvas element) as fingerprinting features [49]. Conceptually, every parameter which is rather stable over time and likely different between users, is a potential fingerprinting feature. This includes exotic features like measuring the dimensions of Unicode glyphs [294], checking for the existence of certain JavaScript functionalities [49, 45], utilizing the drift from exact time in TCP timestamps [180], checking the leakage of the battery status [182] or retrieving a partial subset of visited websites [75]. However, these advanced approaches have limited reliability (e.g., due to network latencies) [295], and are mostly not used in the wild.

To protect the user against fingerprinting, a few tools [289, 179, 296, 297] were proposed, trying to hide unique browser features by randomizing, blocking or deactivating them. Unfortunately, there methods lack usability when browsing websites due to blocked features, they are detectable by the fingerprinter [30, 59, 75, 46, 77] due to unrealistic features, or have flaws making users still identifiable. We expose these issues in Section 6.1.

In this chapter, we present a robust approach to protect a user against browser fingerprinting, implemented in our *Disguised Chromium Browser* (DCB), published in [71]. Instead of disabling or randomizing system and browser parameters, we use a large set of real world parameters that are fixed for the entire browsing session and change automatically for the next session. This prevents detectability through unrealistic or constantly changing system parameters, if a fingerprinter requests a parameter multiple times. In addition, we are the first to effectively protect against Flash as well as canvas fingerprinting without completely deactivating these features. This significantly enhances the usability of web browsers with integrated anti-fingerprinting strategies. Through a thorough study of canvas fingerprinting, we developed a novel and deterministic approach to prevent canvas fingerprinting based on transparently modifying the canvas element in every browsing session. In contrast to other approaches, our solution is not based on adding patterns or noise to the canvas output. Because the canvas element is never unique, the fingerprinter is not able to detect our modifications and is not able to re-identify the user. We show the robustness of the algorithm and demonstrate that every generated canvas element is unique. In contrast to other anti-fingerprinting tools, we also implement a new protection mechanism against the retrieval of system fonts via Flash.

Finally, we evaluate our solution against real world fingerprinting tools and demonstrate its effective protection against fingerprinting by creating unique fingerprints in over 99% of 70.000 browser sessions. We show that fingerprinters cannot notice the presence of our counter-fingerprinting techniques due to enhanced protection mechanisms inside the browser itself and the usage of real world parameters.

6.1 Related Work

Fingerprinting Features

Various research has been conducted on browser fingerprinting features as well as user protection [53]. To obtain an extensive and up to date list of fingerprinting features, we analyzed popular fingerprinting scripts such as *FingerprintJS*[1]*, Coinbase Payment Button*[2]*,* and *BlueCava*[3] as well as data gathered by other researchers [46, 12, 45, 52, 30, 50, 179, 298]. The following list contains popular fingerprinting features that are retrievable through JavaScript, Flash, CSS or the HTTP header [53]:

System information: Device id, operating system (version, architecture, kernel), screen (resolution, height, width), color depth, pixel depth, timezone, system fonts, system language, date and time, CPU type, clock skew, battery status, mouse movement, keyboard, accelerometer, multitouch capability, microphone, camera, audio capabilities.

Browser information: Browser (version, vendor), User Agent, navigator object, installed plug-ins, preferred and accepted languages, HTTP headers, cookies enabled parameter, supported MIME types, browser history, do-not-track, HTML canvas element, JavaScript runtime, CSS features (font probing, display, etc.).

Network information: IP address, geographic location, TCP timestamps, TCP/IP parameters, proxy-piercing.

Flash Capability Class: Version, manufacturer, serverString, language, screenDPI[4].

Canvas: HTML 5 provides a canvas element that can be used for drawing / rendering 2D graphics and to inspect the image data with pixel accuracy via JavaScript. In order to use the canvas element as fingerprint, typically a fingerprinter first renders a defined text using the function *fillText()* or *strokeText()*. Subsequently, the fingerprinter inspects the unique rendering output using the function *getImageData()*, containing the RGBA values for every pixel. Similarly, by using the function *toDataURL()*, a Base64 encoding of the PNG image containing the entire contents of the canvas can be obtained. The unique fingerprint is then produced by hashing the extracted pixel data. Because different graphic cards and rendering engines produce slightly different (but stable) outputs, fingerprinters routinely make use of this element [75, 57].

Fingerprinting Protection

In the following we discuss prominent anti-fingerprinting tools and show their weaknesses which we were able to overcome within our proposed *Disguised Chromium Browser* (DCB).

The **Tor Browser** [289] implements several countermeasures against browser fingerprinting [59], while focusing on anonymity rather than usability. The anti-fingerprinting techniques rely on disabling specific browser features like plug-ins and the canvas element entirely, resulting in a limited usability and web experience (while those features can still be activated by the user to display common web content like Flash, the protection methods become ineffective). Note that even with activated fingerprinting protection the usage of Tor itself is detectable by fingerprinters [30, 59].

In addition, our experiments confirm that Tor's font probing fingerprint protection can still be circumvented [299] by using many dynamically generated iframes. In summary, the Tor Browser can not effectively protect the user against fingerprinting. In contrast to Tor's strategy, our solution uses a large set of real word data to substitute original browser features; furthermore we manipulate Flash and canvas outputs, rather than disabling these functionalities altogether.

FireGloves [51] is a Firefox browser extension that disables access to specific JavaScript objects (like *navigator.plugins*) instead of disabling browser plug-ins entirely. However, an empty list of plug-ins might be used as fingerprinting feature, since this is not a common behavior. Moreover, plug-ins can still be

[1] https://valve.github.io/fingerprintjs

[2] https://www.coinbase.com/docs/merchant_tools/payment_buttons

[3] http://bluecava.com/opt-out/

[4] http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/flash/system/Capabilities.html

instantiated and detected. In comparison, DCB performs a smart manipulation of the plug-ins minor version number in order to guarantee plug-in functionality while preventing fingerprinting. Further, FireGloves allows automatic randomization of specific browser settings. This strategy can be detected by fingerprinters through constant changes to (potentially) unrealistic random settings. Similar to the font probing prevention of the Tor Browser, FireGloves limits the number of fonts retrievable by a website – an approach which can be easily circumvented (see above).

In addition, other font fingerprinting methods like Flash and canvas font probing are not covered. In contrast, DCB manipulates the browser internal font handling itself, instead of limiting the font access. Another drawback of FireGloves is the fact that it is a browser extension, and needs to manipulate JavaScript functions instead of directly accessing browser functionalities. This behavior can be detected and potentially circumvented [30, 179]: A fingerprinter might use *Object:getOwnPropertyDescriptor* to check for manipulated JavaScript objects. To avoid this, we implemented DCB directly in the browser.

PriVaricator [179] is a modified Chromium browser that tries to prevent fingerprinting by randomly changing the browser properties whenever they are queried. For example, PriVaricator returns a random subset of the actual plug-in list by filtering out single entries from the *navigator.plugin* property. Nevertheless, since these plug-ins can still be instantiated, it is possible to detect them. Further, PriVaricator randomly manipulates the properties *offsetHeight*, *offsetWidth*, and the function *getBoundingClientRect()* to prevent font probing. Unfortunately, this implementation is detectable by simply checking for deviations in the output of consecutive requests to the same functions. Additionally, PriVaricator provides no means against the retrieval of system fonts using Flash or the HTML 5 canvas element.

FPGuard [296] is a combination of a browser extension and a customized version of Chromium for detecting and preventing browser fingerprinting at runtime. It uses different hard coded heuristics for detecting browser fingerprinting activities on each website separately. The user can then decide to block or randomize the output of the fingerprintable feature on untrustworthy websites.

Unfortunately, the authors do not go into detail on how the fingerprinting features are randomized. More importantly, as shown in [46, 30], simple randomization will increase a user's identifiability, as unrealistic values will occur. Furthermore, as studies have shown [300], users are not able to decide whether a website is trustworthy or not and will tend to trust a website, so that fingerprinting protection will likely be disabled by users. In comparison, our solution automatically randomizes fingerprintable settings with a set of realistic values.

As another feature, FPGuard randomizes HTML5 canvas images by adding slight noise before the image is read out. We assume that the image manipulation done by FPGuard is non-deterministic. As stated in previous research [49, 52], this approach is detectable. Fingerprinters can create two identical canvas objects and check for differences in the generated image data. Our solution uses a deterministic and transparent algorithm to manipulate canvas rendering itself. This makes it undetectable to fingerprinters, since the output will be the same for the entire browsing-session and different in the next.

In order to prevent font enumeration, FPGuard randomly hides fonts once a certain number of fonts has been loaded. This approach is also non-deterministic, and can be detected by a fingerprinter by checking for the existence of a font through several independent requests. Flash based font enumeration is only blocked within FPGuard by disabling Flash. This not only reduces the usability of websites, but the absence of Flash can be used as fingerprintable information [49]. Moreover, if the fingerprinter circumvents the above mentioned hardcoded thresholds by requesting only a few browser features through a set of dynamically generated iframes, FPGuard will not be able to provide any protection. In contrast, our solution is automatically applied to the fingerprintable features without disabling any browser capabilities or loss of usability.

FPBlock [297] is another anti-fingerprinting tool intercepting a set of HTTP / JavaScript requests and modifying or blocking requested potential fingerprinting features. The goal of the tool is only to stop third-party, cross-website fingerprinting by returning the same modified browser features saved for the this website, whenever the site is visited again. The main disadvantage of FPBlock is its detectability, since fingerprinters will notice the changing browser features trough subsequently visited websites.

Taking our discussion on canvas fingerprinting in Section 6.3 into account, the canvas fingerprinting solution employed in FPBlock is also potentially detectable, since it only adds random noise to the canvas element.

CanvasFingerprintBlock [301] is a Google Chrome extension that prevents canvas fingerprinting by modifying the functions *toDataURL()* and *getImageData()* to only return an empty image when called. Here, the fingerprint of a canvas element will always by the same, hiding system-specific quirks. Unfortunately, this approach decreases the usability of common websites, and increases the potential identifiability of a user, since the output is always unique. Our implementation does not manipulate the functions that are used to retrieve the image data from the canvas. We chose a transparent and undetectable randomization of the image itself, which does not impair the user experience.

Summary

Various existing anti-fingerprinting tools try to protect the user against fingerprinting tracking using different approaches. Unfortunately, all of them only focus on specific aspects fingerprinting tools use to uniquely identify the user. For example, FireGloves and PriVaricatorby randomize just browser settings without blocking font probing, making it possible to re-identify the user. In addition, the anti-fingerprinting tools like the Tor Browser or FPGuard disable important websites features like flash or canvas, making websites unusable. With the proposed DCB we tackle those disadvantages and demonstrate an anti-fingerprinting tool that provides user protection without deactivating website features.

6.2 Anti-Fingerprinting: Disguised Chromium Browser

In order to counter fingerprinting and prohibit re-identification of users, two main strategies can be employed: (1) hide fingerprintable features in order to make all users look the same, such as propagated by Tor, and (2) hide the original features through randomization, like done by FireGloves or PriVaricator.

In our anti-fingerprinting research we identified the advantages of both strategies. On this basis we enhanced the strategies by new protection mechanisms, and implemented them in one *Disguised Chromium Browser* (DCB). In this section, we first present the architecture of DCB and subsequently describe our strategies, the implementation, and the operation of DCB.

6.2.1 Architecture and Implementation

We implemented DCB directly in the Chromium browser version 34.0.1847.131 [302] because of three reasons: First, to avoiding the detection of fingerprinting countermeasures through blocked or intercepted JavaScript CallBacks [30, 49]. Second, to bind all fingerprinting protections in one tool. Third, to provide better performance and browser speed.

We decided against modifying the Tor Browser [303], since it applies certain countermeasures against fingerprinting that might impede with our strategies. To tackle the effect of other tracking mechanisms like cookies [53], we implemented our strategies to use the private browsing mode of Chromium. The required effort to port our implementation to newer versions of Chromium mainly depends on the changes that have been applied in Chromium's source code in the meantime. Compiling our modified Chromium browser takes a few minutes. Once updated and compiled, there is no performance loss during runtime.

DCB relies on a client-server architecture, where a server-side algorithm maintains a database of real world fingerprinting features to enforce a robust browser configuration on the client. On the client side, DCB applies the configuration selected by the server, together with the implemented Flash and canvas fingerprinting protection. DCB allows two strategies for distribution of browser configurations to the client:

N:1 – Many Browsers, One Configuration:

Since fingerprinters collect a multitude of fingerprinting features, the likelihood of several browsers sharing the same system properties is considerably low. For that reason, when fingerprinters observe the same fingerprint more than once, they most likely will assume the fingerprint to belong to the same browser as in a previous observation. The $N : 1$ strategy aims at configuring as many browsers (N) as possible to share the same configuration, making all those browsers look the same to the fingerprinter. This decreases the surprisal of observing a specific configuration and limits the re-identification of one specific user.

1:N – One Browser, Many Configurations:

In the $1 : N$ strategy, the actual browser and system configuration is hidden to the outer world in order to prevent re-identification by fingerprinters. DCB in $1 : N$ mode constantly changes the browser configuration on each start. Compared to existing tools like PriVaricator [179] or FPGuard [296], which randomly change fingerprintable parameters, our approach randomizes features to realistic values and prevents constantly changing system parameters. This limits detectability.

Configuration Server:

Every browser has a configuration (its original), consisting of single-value features (like the Windows version), and of multi-value features (such as a list of fonts). The configuration server is responsible to store such real world system and browser properties (fingerprinting features) and to generate or assign configurations to clients based on the selected strategy. To ensure a realistic randomization of features, we use a pre-stored data set of 23,709 real-world fingerprinting features. These anonymized features were gathered in a large scale study [304] similar to Panopticlick [46]. Furthermore, upon DCB startup, clients send their initial configuration to the server, which is added to the database.

DCB can work either with a local or a global configuration server. To guarantee trust, configuration servers may be operated by trustworthy organizations such as the CCC[5] or EFF[6]. Tackling potential single-point-of-failure problems, the amount of servers needs to be raised while the user base grows. To avoid traceability and privacy problems, the server never saves any connection details such as the IP address of users, and can be verified due to the open source implementation.

We implemented the configuration server using CakePHP[7], since it is an easily expandable, model-view-control orientated framework. The upload of a client's configuration is done via HTTPS. Once the client identifies a configuration change, the new configuration will be sent to the sever.

Configuration Groups:

It is important to pay attention to inconsistencies and usability constraints when changing or hiding system and browser properties. For example, if a specific operating system (OS) is propagated, the list of plug-ins reported should match the OS. We decided to use groups of configurations sharing similar values, so that no browser adapts a configuration that contradicts its actual system configuration. One specific configuration group could consist of all users using browsers on the same OS and language. Here, all browsers in this configuration group will only get those configurations (e.g., list of plug-ins), that are available for the given OS. Because a large number of configuration groups is counterproductive in hiding the user effectively, we predefined configuration groups for the user's language and different Windows versions (like Windows Version NT6.1 & English or Windows Version NT5.0 & French). The fewer groups exist, the smaller the surprisal observing any browser.

Depending on the actual system and therefore the configuration group, the $N : 1$ strategy aims to set all fingerprintable features to the most frequent parameters in this configuration group, fitting with the user's environment. This guarantees robustness and usability. At the moment of development the

[5] `https://www.ccc.de`

[6] `https://www.eff.org`

[7] `http://cakephp.org`

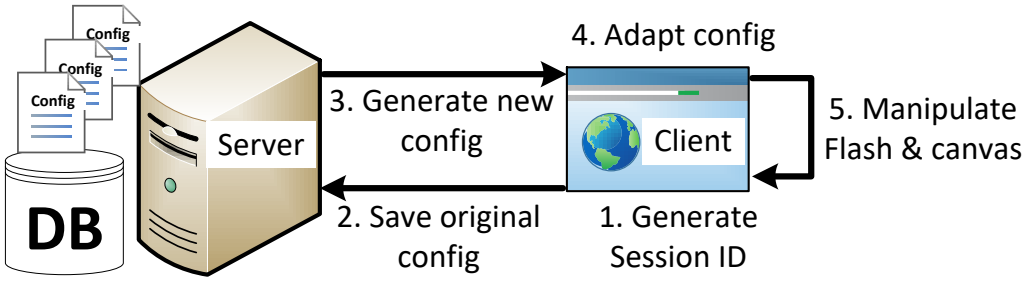

Figure 6.1: DCB: Initialization of a browser session and mode of operation.

configuration contained the following parameters: screen information, browser language, user agent (Chrome & WebKit/Blink version, Windows architecture), time and date, system fonts and plug-ins. Further, we use the most common intersection of fonts and plug-ins of all browsers in the group.

6.2.2 DCB: Mode of Operation

Once a new browser session is started, several steps are performed, shown in Figure 6.1. First, DCB generates a random session identifier ID (step 1). The ID is used as input in the canvas fingerprinting protection algorithm as well as an identifier for the browsing session. In the second step, DCB collects configuration information on the client (such as fonts, User Agent, or plug-ins) into a JSON encoded configuration file and sends it to the server.

If the $N : 1$ strategy is selected, the configuration server checks if there exist a *configuration group* of clients that share similar features. If not, the server creates a new group that fits to the browser's configuration. Otherwise, if a browser fits into an existing group, the group's configuration is revised and if necessary updated. This may be necessary to adjust the property of a feature to the majority of users' configurations. Subsequently, the server returns the selected configuration, encoded in JSON, to the DCB browser (step 3). This strategy assures that the generated group configuration is based on the most frequent, already stored configuration data, and the best fitting configuration group for the client (depending on general system features). The browser then adapts the new configuration as its own (step 4), and manipulates the DLL file of Adobe Flash Player (step 5) in order to change propagated values such as the screen resolution or operating system. A full description on the general manipulation of Flash and system fonts are presented in Section 6.2.3. Specific manipulations to the fingerprinting features according to the $N : 1$ strategy are detailed in Section 6.2.4.

If the $1 : N$ strategy is used, the server responds with a random configuration taken from the database. This configuration is encoded in JSON and subsequently sent to the browser (step 3). Steps 4 and 5 are equivalent to the procedure described for $N : 1$. Since DCB modifies browser and Flash equally, a fingerprinter will see the same parameters when settings are retrieved by JavaScript or Flash. Finally, changes are fixed for the entire browsing session and reset automatically for the next session. Details on how fingerprinting features are manipulated in the $1 : N$ case are given in Section 6.2.5.

Note that a server is needed to gather and generate accurate configurations for both strategies. The server aides in adapting one configuration for a set of users ($1 : N$); in addition it stores realistic (original) configurations for the $N : 1$ strategy.

6.2.3 Flash and System Font Protection Mechanisms

DCB enables a general Flash and system font fingerprinting protection for both $N : 1$ and $1 : N$ strategies. In the following, we present the details on the implementation.

Flash Player Capabilities-Class: This class provides a wide range of fingerprintable system information such as screen information or Windows version, which are not covered by other anti-fingerprinting tools when Flash is activated in the browser. The Capabilities functionality is compiled and encoded as hex strings inside the *pepperflash.dll* file. To ensure fingerprinting protection we therefore use Python to manipulate the Capabilities-Class on all system parameters that are accessible by Flash. The parameters are changed to match the values that are retrievable by JavaScript (like User Agent, screen object or others), and were already modified by the DCB configuration.
System Fonts: To counter font probing via CSS and JavaScript, we internally change requested font names either to existing fonts or non-existing fonts, depending on the strategy, whether we want to fake the existence or non-existence of a system font. For example, we might hide the existence of the font *Comic Sans* by changing the request into a fictive font '*aaa*'. Internally, the font will not be found and therefore the browser's default fallback font will be used, even the font is actually installed on the system. Analogously, to fake the existence of a non-installed font, DCB automatically renders the text with another existing system font such as *Comic Sans*. Since *Comic Sans* is not the fallback font, the fingerprinter will then assume the existence of the requested font. We also manipulate the list of fonts returned by Flash. Here, we either add new font names or filter out existing names from Chromium's internal font list (see Sections 6.2.4 and 6.2.5 for a detailed description).

6.2.4 Specific N:1 Implementation

In this section we describe the specific method how fingerprinting features are manipulated in the $N:1$ strategy.

Screen Information: In order to achieve a common screen configuration, the server will determine the most frequent screen resolution, color depth, and pixel depth for every configuration group. The server will also choose among those information that fit with the available users' screen (monitor).
Browser Language: Browsers using the same main language and operating system are joined into a configuration group. All other languages that are part of the *HTTP_ACCEPT_LANGUAGE* header commonly have a lower priority (q value) than the main language. We adapt a language with a lower priority only to a configuration group in case that at least 3/4 of all browsers in the configuration group share that common language.
Plug-in Information: Since we want to create configuration groups that are as large as possible, our goal for the $N:1$ strategy is to reduce the revealing of information to a minimum, making it easier to join browsers into anonymity groups. For this reason we disable all plug-ins that are not part of every browser of a configuration group, which consequently can result in disabling all plug-ins that are not shipped with Chromium. Along with this step, the most common name and description of a plug-in is adopted in order to equalize the possible version information. Note that we could have considered plug-ins as usability constraint and therefore avoided the chance of disabling any plug-ins. However, as this would reduce the size of the configuration groups we decided against it. Yet, anti-fingerprinting tools with a large user base could choose to handle plug-ins as constraint feature.
User Agent: For every configuration group the most common Chrome version, WebKit/Blink version and Windows architecture are selected and then shared between all browsers of that group.
System Fonts: The list of fonts is set to the common intersection of fonts of all browsers in the configuration group. To share and apply the list of fonts among the browsers of a configuration group, we use the fact that the Windows version will be the same for all browsers sharing the same configuration. Here, the final font configuration only contains the fonts delivered with the initial operating system installation.
Time and Date: In order to set a common time zone offset for every browser of a configuration group, the most frequent offset of all browsers in group is selected.

6.2.5 Specific 1:N Implementation

DCB applies the following fingerprint feature modifications in the $1 : N$ strategy to achieve the goal of a diverse browser configuration.

Screen Information: The screen resolution, color depth, and pixel depth, are randomly selected among the already observed (pre-stored) values. Regarding the available resolution, the height of the selected screen resolution is reduced according to the height of the taskbar of the selected Windows version. For example in Windows 7 the taskbar might have a height of 30 or 40 pixels depending on the user choice. If more than one value is possible, the height is chosen randomly.
Browser Language: While Flash and *navigator.language* only return the main language of the user, we do not need to change these values, as we keep the main language for usability reasons. We only manipulate the list of additional languages returned by *HTTP_ACCEPT_LANGUAGE*. Languages are separated by a comma, the language code and priority q are separated by a semicolon. Note that the priority of the main language with the highest priority of 1 is omitted in the language header. Besides the main language and every language with a priority $\geq$0.8, we randomly select up to three additional languages with a random quality between 0.7 and 0.1 (steps of 0.1). We only add language codes according to language tag detection standard *BCP 47* [305] in order to prevent implausible language codes. An exemplary language acceptance header and the priority of the corresponding languages looks like: de-DE,de;q=0.8,en-US;q=0.6,en;q=0.4
Plug-in Information: If at least 5 different versions of a plug-in are available in the database, the server will select a plug-in description randomly. Otherwise, the server searches for version numbers in the name and description and manipulates the minor version numbers. If the minor version is not available, we randomly change the provided version number.
User Agent: We do not hide the actual browser vendor, as it could be detected using vendor specific features [30]. Instead, we manipulate the version numbers according to real browser which provides the necessary fingerprint diversity. The WebKit/Blink and Chrome version numbers are changed according to the algorithm used to manipulate the plug-in version numbers (described above). The Windows version is randomly chosen among the already observed and pre-stored Windows versions. The Windows architecture is chosen from the fixed list of possible, vendor specific values, which depend on the processor and the bit version of the operating system.
System Fonts: The list of fonts contains 90 to 320 randomly chosen fonts from the list of previously observed fonts (i.e., from the configuration server), along with the fonts that are shipped with the specific Windows version. Therefore, only realistic font names will be observed by a fingerprinter.
Time and Date: The function *getTimezoneOffset()* is commonly used as a fingerprinting feature. In order to maintain consistency, we change the time zone offset along with all other time and date information retrievable by the *Date* class.

6.3 Canvas Anti-Fingerprinting

Existing canvas anti-fingerprinting tools manipulate the canvas readout functions *toDataURL()* and *getImageData()* so that random noise is added in order to prevent fingerprinting. As researchers noted [75], it is not suffice to add random noise whenever image data is requested from the canvas. Fingerprinters can detect this noise through subsequent identical function calls and comparing the results. Various strategies could be applied to counter the detection of modifying *getImageData()* or *toDataURL()*, but we argue that any approach would face the common problem of detectability (see the detailed description in Section 6.3.1). In contrast, in our approach we modify the rendering of the canvas itself, and always perform the same modification for the entire browsing session.

6.3.1 Weaknesses in Counter Detection Strategies of Canvas Manipulation

Although the below described strategies could be applied to hide the manipulation of *toDataURL()* or *getImageData()*, the weaknesses that may lead to their detection still exist:

(1) One may store hashes of previously generated canvases along with their modified version for the duration of a browser session. As soon as an image with the same hash is requested, the previous modification of the image is returned to prevent the detection of differences. Yet, this approach would be detectable if a fingerprinter would add a localized change to the canvas and only compare non-effected parts to a previously returned output. Since this approach calculates the hash of the overall image, a new modification is applied causing the hash to change, unmasking the anti-fingerprinting measures.
(2) To circumvent the comparison of localized changes, an anti-fingerprinting algorithm could modify the localized changes and copy the old modification of those parts that are exactly the same. Still, the problem exists if many different changes are added and the fingerprinter would compare a partial hash of those areas.
(3) In order to avoid the detectability of small modifications, an algorithm could store all distinct canvases of a session. When a new canvas is about to be manipulated, all areas of similarity of prior canvases need to be re-placed with their respective recorded images. Newly observed areas would be then modified separately with random noise. Again, this approach could be detectable when a fingerprinting script would render the same image information twice on one image. If the image was new to the algorithm, it would randomize both parts differently.

6.3.2 Robust Canvas Fingerprinting Protection

Instead of randomly manipulating the canvas readout functions *toDataURL()* or *getImageData()*, DCB deterministically changes the canvas rendering function *CanvasRenderingContext2D::drawTextInternal()* directly for each browsing session. This function is using *fillText()* and *strokeText()*, which covers all known canvas fingerprinting approaches. Moreover, the approach can be applied to new rendering functions used for fingerprinting.

We use the random session identifier, generated at Chromium startup, to steer the modifications. Due to the randomness of the session identifier, it is guaranteed that *fillText()* returns deterministic values during a browser session but different ones in subsequent browser sessions. Figure 6.2 illustrates the process. When the internal method for handling *fillText()* requests is called, we first backup the image buffer (step 1), and wait until the function has finished rendering text (as part of the fingerprinting process) into the buffer (step 2). In the next step we compare the previously saved image data with the new image data in the buffer and store the positions of every pixel that has changed (step 3). In the next step we apply the image manipulation algorithm (step 4) as described in the next paragraph. Finally, the image data is returned (step 5).

Figure 6.2: Canvas processing algorithm.

6.3.3 Image manipulation algorithm

The algorithm is implemented in C++, since native functions will render canvas elements faster[8]. The image manipulation algorithm works on a per-pixel basis, and is only applied to pixels that are close to a color border; i.e., to pixels where its top, right, left, and bottom neighbor do not share the same values. We encode a pixel p as a triple (r, g, b, a), where r, b, g, a are the red, green, blue, and alpha values $r, g, b, a \in \{0,..., 255\}$. We modify p to p' by adding an offset to r, g, b. We first concatenate r, g, b and a with the pseudo random session identifier s generated at startup, apply the SHA256 hash, and call the result t. The offset is computed by taking subsequent blocks of 20 characters from t modulo a constant c, which is by default set to 3. Then, for each r, g, b we either add or subtract the offset depending on whether their value was above or below 128, resulting in p'.

Note that the image manipulation is deterministic and not visible to the user (using the default setting $c = 3$). Therefore, it is not possible for fingerprinters to detect this strategy and to remove or subtract any modification from the canvas to reconstruct the original image, like suggested in [78]. Using high values for c, for example $c = 50$, is not necessary, as it does not increase the effectiveness of the algorithm and will only make the changes more visible. Assuming a canvas with 1806 pixels, containing a total of 1030 different colors, and using the standard value of $c = 3$, we already get $3^{1030} \approx 2.722 \times 10^{491}$ possible combinations in changing the pixel values. As shown in Section 6.4.1, this is enough to prevent fingerprinting.

Mathematic description of the Canvas image manipulation algorithm:

$c = 3$
$p \in P, P = (r, g, b, a), r, g, b, a \in x | 0 \leq x \leq 255$
$s \in S, S = (c_i)_{i=1}^{64}, c_i \in 0, 1, ...9 \cup 'A', 'B', ..., 'F'$
$SHA256(x) : (x_i)_{i=0}^{n} \rightarrow S$
$Hex2Dex(x) : 0, 1, ...9 \cup A, B, ...F \rightarrow 0, 1..., 15$
$i)$
$t = SHA256(r_1, r_2, r_3, g_1, g_2, g_3, b_1, b_2, b_3, s_1, s_2, ..., s_{64}),$
$r_1 = \lfloor \frac{r}{100} \rfloor, r_2 = \lfloor \frac{r - r_1 \times 100}{10} \rfloor, r_3 = \lfloor r - 100r_1 - 10r_2 \rfloor$
$g_1 = \lfloor \frac{g}{100} \rfloor, g_2 = \lfloor \frac{g - g_1 \times 100}{10} \rfloor, g_3 = \lfloor g - 100g_1 - 10g_2 \rfloor$
$b_1 = \lfloor \frac{b}{100} \rfloor, b_2 = \lfloor \frac{b - b_1 \times 100}{10} \rfloor, b_3 = \lfloor b - 100r_1 - 10b_2 \rfloor$
$ii)$
$r_{offset} = (\sum_{i=1}^{20}(t_i)) mod c$
$g_{offset} = (\sum_{i=21}^{40}(t_i)) mod c$
$b_{offset} = (\sum_{i=41}^{60}(t_i)) mod c$
$iii)$
$$r' = \begin{cases} r + r_{offset}, & r < 128 \\ r - r_{offset}, & r \geq 128 \end{cases} \quad g' = \begin{cases} g + g_{offset}, & g < 128 \\ g - g_{offset}, & g \geq 128 \end{cases}$$
$$b' = \begin{cases} b + b_{offset}, & b < 128 \\ b - b_{offset}, & b \geq 128 \end{cases}$$
$p' = (r', g', b', a)$

6.4 Evaluation

We tested DCB with both strategies $N : 1$ (many browser, one configuration) and $1 : N$ (one browser, many configurations) for its effectiveness against real world fingerprinters. In addition, we also evalu-

[8] A JavaScript implementation would also be possible. However, as noted before, JavaScript CallBacks are detectable and can potentially be circumvented.

ated our canvas anti-fingerprinting strategy independently from the other features to show the robustness and practicality of our approach.

6.4.1 1:N – One Browser, Many Configurations

The goal of the $1 : N$ strategy is to establish on each browser startup a completely new configuration (fingerprint) so that fingerprinters will not be able to re-identify the user in the next browser session. In addition, the strategy shall guarantee realistic settings and an unchanged user experience on the web. We examined the effectiveness of DCB running in $1 : N$ mode by analyzing the reported fingerprint of three popular fingerprinters: *FingerprintJS*, *Coinbase Payment Button*, and *BlueCava*.

Experimental Set-Up:

We evaluated DCB under realistic conditions on a standard Windows PC. To automatically simulate user behavior and the generation of a new (modified) configuration by DCB, we started the browser 10,000 times on specific web pages (described next) for each fingerprinter. In every session and for every fingerprinter we saved and compared the generated fingerprint.

FingerprintJS is an open source fingerprinting library written in JavaScript. For retrieving FingerprintJS fingerprints we created a web page that sends the fingerprint to a PHP script via an AJAX request. To derive the fingerprint, we chose three fingerprinting feature sets FS1-FS3 that were present in an exemplary HTML file delivered along with the library. Feature set FS1 includes the features: User Agent, navigator language, color depth, time zone offset, plug-in list, *this.hasSessionStorage()*, *this.hasLocalStorage()*, *window.indexedDB*, *typeof document.body.addBehavior*, *typeof undefined*, *typeof window.openDatabase*, *navigator.cpuClass*, *navigator.platform*, *navigator.doNotTrack*. FS2 uses the same fingerprinting features along with canvas information. FS3 adds the screen resolution to the fingerprint of FS1.

Coinbase generates a fingerprint on the client via JavaScript (its intended use is to protect Bitcoin payments against fraud). We identified two functions, *browserAnalytics()* and *fingerprint()*, that Coinbase uses to retrieve various information[9] about a user's computer. While *browserAnalytics()* returns the information in plaintext, *fingerprint()* creates the actual fingerprint by hashing the collected information using MD5. In order to compare the values of *browserAnalytics()*, we also hash them with MD5 and store them along with the value returned by *fingerprint()* in the database. For this we isolated the fingerprinting code from the Coinbase website and applied it on our own website.

BlueCava is a top 5 fingerprinter [179] that calculates its fingerprint on the server side using a large quantity of fingerprinting features[10]. However, it provides an opt-out page where the fingerprint of the browser can be retrieved. We make use of this feature in our test and grab the fingerprint in an automated manner by using JavaScript.

Fingerprint Robustness of BlueCava:

Eckersley [46] suggested that a fingerprinter might easily compensate changes in some of the used features. In order to evaluate the effectiveness of our strategies, we first analyzed the robustness of fingerprints against manual configuration changes. For this experiment we used BlueCava, since it scans a large set of fingerprinting features.

We set up a virtual machine running Windows 7 and installed Google Chrome version 34. We then installed a set of 2000 fonts (BlueCava uses font probing with Flash and JavaScript [30]) and started Chrome in private mode to retrieve the fingerprint from BlueCava's opt-out page. To estimate the robustness to changing fingerprint features, we compared the fingerprints on random system font changes, by adding 5 fonts, then deleting 173 fonts. In all cases the fingerprint was identical. The fingerprint

[9] `https://coinbase.com/assets/application.js`
[10] `http://ds.bluecava.com/v50/AC/BCAC5.js`

Table 6.1: Resulting fingerprints on evaluating 1:N strategy in 10,000 browser sessions [higher is better], and N:1 on 12 systems [lower is better].

Fingerprints in strategy	Bluecava	FingerprintJS			Coinbase		Canvas script
		FS1	FS2	FS3	FP1	FP2	
1:N	10,000	9,910	10,000	9,992	10,000	10,000	10,000
N:1	1	1	2	1	7	8	N/A

changed after deleting all fonts apart from the standard system fonts and additionally disabling four of Chrome's standard plug-ins (not Flash). This indicates that BlueCava's fingerprint seems to be robust towards changes of a user's system configuration, making it a good benchmark to evaluate the effectiveness of our $1 : N$ strategy.

Canvas Fingerprinting Set-Up:

We implemented our countermeasures against canvas fingerprinting in conjunction with the $1 : N$ strategy. We isolated the canvas fingerprinting script of AddThis[11] (a popular fingerprinter [57]), implemented it on our website and examined the reported fingerprints in an extensive study. In a run of 10,000 browser sessions we stored every generated fingerprinting canvas image as MD5 hash in our database. To create the rendered image, the canvas fingerprinting function uses *fillText()*, rendering text with the fallback font.

Results

Table 6.1 presents the effectiveness of our $1 : N$ strategy in creating distinct fingerprints. For both BlueCava and Coinbase, every fingerprint in the set of 10,000 fingerprints of different DCB browser sessions turned out to be unique. As the fingerprint change protection of BlueCava overcame the course of all 10,000 browser sessions, it would have been impossible for BlueCava to track DCB users over consecutive browser sessions.

On FingerprintJS we recorded 99 duplicates on two out of three tested feature sets. For the features set FS1, 88 fingerprints occurred twice and one fingerprint occurred three times. We explain this behavior by the small amount of features that FingerprintJS collects to generate the fingerprint. FS3, which is an extension of FS1, additionally using the computer's screen resolution, produced only nine duplicates, as it uses more fingerprinting features that can be modified by DCB. In practice, if a fingerprinter uses a smaller fingerprint feature set, the fingerprint becomes less reliable and also less unique, as can be seen in column FS1 and FS3. Note that FingerprintJS collects no information about the system fonts, which normally contributes greatly to a fingerprint's entropy and would otherwise produce a unique fingerprint like for FS2.

Further, FS2, which adds HTML 5 canvas information to FS1's fingerprinted information, resulted in a set of 10,000 unique fingerprints. This is due to our canvas fingerprinting protection mechanism described in Section 6.3.2. In order to assess the effectives of our strategy against canvas fingerprinting we also applied a separate test run of canvas specific fingerprinting as described in Section 6.4.1.

It turned out that every single fingerprinting image in our set of 10,000 browser sessions was unique (Table 6.1, column Canvas). Therefore, our strategy against canvas fingerprinting is highly effective, as duplicate fingerprints are nearly impossible (see Section 6.3.2).

Preliminary Conclusion:

The non-commercial open source library FingerprintJS produced over 99% unique fingerpints in 10,000 browser sessions. The commercial fingerprinting scripts of Coinbase and BlueCava produced

[11] `http://ct1.addthis.com/static/r07/core130.js`

100% unique fingerprints in the run of 10,000 browser sessions. More important, even though BlueCava is applying means to compensate for fingerprint changes, our $1 : N$ strategy produced changes that were large enough to break out of these compensation mechanisms. Therefore, we can conclude that our $1 : N$ strategy is highly effective against long-term traceability, as a fingerprint cannot be used to identify a user over consecutive browser sessions. In addition, we ensure website usability through automatic Flash and canvas anti-fingerprinting protection mechanisms without disabling those features. We also manually verified the perfect functionality of the top 20 ranked *Alxa.com* websites.

6.4.2 N:1 – Many Browsers, One Configuration

The $N : 1$ strategy aims at decreasing the surprisal of observing a browser instance by increasing the number of browsers sharing the same configuration. Therefore, evaluating this strategy required the execution of DCB on several systems with varying configurations. As above, we tested the $N : 1$ strategy by matching fingerprints generated by the three introduced fingerprinters.

Experimental Set-Up:

We installed our DCB on ten virtual machines (VM) and on two physical computers as a reference. Since we wanted to guarantee that these systems shared the same configuration group, we configured all systems to have the same operating system (Windows 7, Home Premium 32 bit or Professional 64 bit with Service Pack 1), and set the same browser language. We deliberately prepared different VMs, by installing various programs, including different office suites, PDF readers and other software. After the installation we observed that up to 105 additional fonts were installed, with an average of 22 fonts per virtual machine, and up to ten new browser plug-ins. In addition, we added another 25 distinct fonts for every VM, chose different screen resolutions, and selected various time zones. Then, we executed DCB in $N : 1$ mode on all 12 systems and tested it separately against each fingerprinter (BlueCava, Coinbase, FingerprintJS) and the canvas fingerprinting script presented in Section 6.4.1.

Results

Table 6.1 summarizes the results. For BlueCava as well as the feature set FS1 and FS3 on FingerprintJS all 12 systems generated the same fingerprint, as intended. FS2, which adds the canvas element, generated one indistinct fingerprint on one of the PCs, since the canvas protection mechanism can not be shared between DCB browsers and is therefore not implemented in $N : 1$. For the Coinbase script our algorithm performed not as good as expected. Only 46% of the systems shared the same fingerprint. This could be explained by the fact that Coinbase applies less sophisticated compensation mechanism against configuration changes.

We can conclude that the $N : 1$ strategy is prone to additional fingerprinting features that might not be covered by the implementation, as additional entropy for distinguishing between otherwise equal configurations might be introduced. Therefore, this requires to constantly consider new fingerprinting features.

6.4.3 Comparison of Anti-Fingerprinting Features

Finally, we show in Table 6.2 the advantages of DCB by comparing it against other anti-fingerprinting tools: Tor Browser [303], FireGloves [51], FPGuard [298], FPBlock [297], PriVaricator [179].

We were able to successfully overcome the weaknesses and flaws of existing anti-fingerprinting tools and additionally implemented new Flash as well as canvas fingerprinting protection. For future work we want to implement further functionality which by now are neither implemented in existing tools, nor used by fingerprinters in the wild (mostly because of to limited reliability and network latency [295]).

Table 6.2: Comparison of fingerprinting feature coverage: PriVaricator (PV), FireGloves (FG), FPGuard (FPG), FPBlock (FPB), Tor, DCB.

Feature	Retrieval	PV	FG	FPG	FPB	Tor	DCB
Browser History	Rend. timing attack	✓	x	?	x	✓	✓
Browser	JS engine	x	x	x	x	x$_0$	x
	navigator.appVersion	x	✓$_1$	✓$_2$	x	✓$_3$	✓
Clock skew	TCP/ICMP	x	x	x	x	✓$_4$	x
Flash Version	Adope Flash	x	x	✓$_5$	✓$_5$	✓$_5$	✓
	JavaScript	x	✓$_1$	x	?	✓$_3$	✓
Fonts	Using Flash	x	x	✓$_5$	✓$_5$	✓$_5$	✓
	Using CSS	✓$_6$	✓$_6$	✓$_6$	✓$_6$	✓$_6$	✓
	HTML5 canvas	x	x	x	x	✓$_6$	✓
IP/Network info	TCP/IP stack	x	x	x	x	✓	x
Language	HTTP header	x	✓$_1$	?	✓	✓$_3$	✓
	JavaScript	x	✓$_1$	?	✓	✓$_3$	✓
Math constants	JavaScript	x	x	x	x	x	x
Mime types	JavaScript	✓$_7$	✓$_7$	✓$_7$	✓$_7$	✓$_7$	✓
Plugins	JavaScript	✓$_7$	✓$_7$	✓$_7$	✓$_7$	✓$_7$	✓
Canvas	JavaScript	x	x	✓$_8$	✓$_8$	✓$_2$	✓
Screen height	JavaScript	x	✓$_1$	✓$_2$	✓	✓$_3$	✓
Screen width	JavaScript	x	✓$_1$	✓$_2$	✓	✓$_3$	✓
Screen color depth	JavaScript	x	x	✓$_2$	✓	✓$_3$	✓
Screen pixel depth	JavaScript	x	x	✓$_2$	✓	✓$_3$	✓
Screen resolution	Adobe Flash	x	x	✓$_5$	✓$_5$	✓$_5$	✓
	JavaScript	x	✓$_1$	✓$_2$	✓	✓$_3$	✓
Time & date	JavaScript	x	x	x	x	✓$_3$	✓
Time zone offset	JavaScript	x	✓$_1$	x	✓	✓$_3$	✓
User Agent	HTTP header	x	✓$_1$	?	✓	✓$_3$	✓
	JavaScript	x	✓$_1$	✓$_2$	✓	✓$_3$	✓
Windows version	Adobe Flash	x	x	✓$_5$	✓$_5$	x$_5$	✓
	User Agent	x	✓$_1$	✓$_2$	✓	✓$_3$	✓

0) Protection only implemented for a subset of functions. 1) Randomization through constantly changing unrealistic system parameters causes detectability. 2) Expects user approval. Inexperienced users might reveal data to trackers. 3) Blocks access or sets parameter to a fixed value. This causes usability constraints and can be used as fingerprintable information. 4) IP address is hidden due to default usage of an anonymity network. 5) Blocks Flash and impedes the usability of websites. Flash can be activated by the user and become a source of fingerprintable information. 6) It only limits font probing. This can be circumvented by using dynamic generated iframs scanning for small amount of fonts. 7) Randomly removes plug-ins or adds non existing plug-ins from/to plug-in list (causing usability constraints and is detectable since plug-ins can still be instantiated). 8) Non-deterministic usage by adding noise limits usability. This makes it detectable by creating multiple canvas outputs in same browsing session by checking for differences.

6.5 Summary

We showed flaws in existing anti-fingerprinting tools and presented new approaches preventing fingerprinting. Futher, we implemented and evaluated the effectiveness of two strategies $N:1$ and $1:N$, which enhance browser/system fingerprinting protection with the first built-in Flash fingerprinting protection

without deactivating Flash. The $N : 1$ strategy aims at decreasing the chance of being uniquely identifiable in a large set of DCB browser instances by using the same browser configuration for all users. The $1 : N$ strategy aims at breaking the fingerprint by applying major browser/system configuration changes using real word properties every time a new browser session is started.

Moreover, we demonstrated the effectiveness of the first robust solution against canvas fingerprinting that, in contrast to other approaches, does neither block nor randomly manipulate any canvas functionality used to retrieve the rendered image data.

7 Conclusion and Future Work

The results of this work show the risks, threats and implications of web tracking. We demonstrate that the usage of personal data gathered by third-parties is not limited to personalized ads. For example, sensitive private user data can also be used to estimate the private credit score, affect the personal health insurance contract, or alter the prize of goods displayed for specific users. Since the personal data are stored on long term, we can not predict all further use cases invading the users personal life. In our description of the web tracking ecosystem, including the market and stakeholder, we present the tracking techniques used in the wild as well as available protection mechanisms along with their weaknesses. By crawling the 1,634 mostly visited websites in Germany for 16 months, our long term tracking analysis shows that trackers are able to follow the users on 90% of the scanned websites. Moreover, we demonstrate that the top trackers such as Google and Facebook are able to gather user data on 25% of the top 40,000 global ranked websites. Tackling this problem, we implemented an automated system to detect and protect against the most common tracking elements used. Those include cookies, web-beacons, JavaScripts, HTML5 LokalStorage, Flash LSO, and iFrames. The list of resulting tracking domains has been used successfully within the ad-blocking feature TPL in the Internet Explorer. In addition, a publicly available website *www.trackyourtracker.de* was set up for many years to provide users an interactive tool showing how many trackers within the 1,634 mostly visited websites in Germany can track the user's online behavior.

Further, we crawled the top 50,000 global ranked websites for several months to evaluate the tracking technique using redirect links. By analyzing 1.2 million redirect traces we were able to classify the top redirect trackers and found that 11.6% domains use redirect tracking. To visualize our findings and make the data publicly available, we implemented an interactive web tool showing the clusters of redirect trackers in a network graph. Furthermore, we compare different tracking classification methods and provide possible protection strategies against redirect tracking. Since tracking elements are placed as first-party on a redirect page, third-party add-ons will not help against this kind of tracking. These contribution fulfilled our first research focus to systematically analyze and classify web tracking.

The evaluation of available tracking protection mechanisms was the second research focus of this thesis. We showed for example, that ad-blocking tools still track users by not blocking certain tracking domains after making paid contracts with trackers. Other protection mechanisms, such as the private browsing mode, are not sufficient to stop tracking, since they still allow trackers to record the users' Internet activity and collect privacy sensitive data throughout the entire browsing session.

In order to prevent the leakage of privacy sensitive user data, our third and fourth research focus was to design a reference architecture as well as develop an automated test environment under real-world conditions to evaluate it's effectiveness. Therefor, we first extended the Chrome browser with our *Site Isolation* feature to locally store the website's state into separate containers. Here, we introduce a novel tracking classification, calculating the entropy of the information content of a website's cookies. We then evaluated our new cookie isolation concept Site Isolation on 1.6 million websites, showing that our prototype can reduce tracking by 44%. Compared to blocking third-party cookies, Site Isolation is more user-friendly and does not depend on the definition of what exactly constitutes a third-party cookie. Within the Site Isolation prototype we also improved the privacy for private browsing and implemented a cookie timeout feature. Moreover, with Site Isolation the user benefits from the security features preventing CORS, CSRF, click-jacking, and limiting the effects of cache-based tracking, cache-timing attacks as well as rendering engine hijacking.

Furthermore, we designed and evaluated in this work a robust approach to protect users against the web tracking technique browser fingerprinting, implemented in our *Disguised Chromium Browser* (DCB). This is the first browser prototype which effectively protects against Flash as well as canvas fingerprinting without completely deactivating these features. We evaluate our solution against real world fingerprint-

ing tools and demonstrate its protection against fingerprinting by creating unique fingerprints in over 99% of 70,000 browser sessions. Comparing our Disguised Chromium Browser to five anti-fingerprinting tools, we show that fingerprinters cannot notice the presence of our counter-fingerprinting techniques due to enhanced protection mechanisms inside the browser itself and the usage of real world parameters.

Concluding our research, we see that protecting the privacy of web users against tracking by just blocking third-party content has become a cat-and-mouse game. Continuously changing tracking methods make it difficult to block all third-party content. On the other hand, it is necessary to accept some third-party content to ensure website functionality. Furthermore, since websites include scripts to detect ad-blocking tools, the users' need to deactivate them to access the website. In this work we present implemented concepts for the automatic isolation of the locally stored website state into separate containers as well as a robust browser fingerprinting protection covering most commonly used tracking techniques. This eliminates the ability of trackers to re-identify users across different sites, by isolating HTTP cookies, HTML5Web Storage, Indexed DB, and the browsing cache.

Future work:

A future research should analyze http traces, embedded (third-party) website elements and JavaScripts for tracking patterns in order to identify new tracking techniques. Here, future work should focus on detection methods based on machine learning algorithms. By improving the crawler with new detection methods, it should be possible to automatically detect tracking elements without any kind of human support. Especially the identification of new browser fingerprinting features is necessary to improve our anti-fingerprinting strategies. As we presented in this work, various browser as well as computer properties can be accesses through the browser, JavaScripts or other plug-ins such as Flash or Java. New browser features or techniques to identify the user for example by keyboard type patterns are useful for trackers to (re-)identify the user and need to be blocked. Furthermore, in order to improve our anti-fingerprinting strategy against font probing, we plan to use more than one strategy in replacing fonts. This makes sure that a fingerprinter cannot check for repetitive font sizes in order to detect the strategy. We could achieve this by using font size data of fonts whose existence has been hidden or by installing an additional set of fonts. Regarding the manipulation of Flash binaries, future research needs to be done implementing our manipulation functionality directly in the browser rather than using an external Python script. In a future study it would also be interesting to examine whether fingerprint databases of fingerprinters aiming at detecting configuration changes could be poisoned by a large set of false data, caused by an intensive usage of our $1 : N$ implementation.

Using our crawling architecture for further long term studies would help to display active tracking techniques. This should point out which methods are most effective and probably less covered by protection mechanisms. In addition, there is a need of evaluating which privacy sensible data leads to more precise personalized ads in real-time bidding. Better insights would help to develop more sophisticated protection mechanisms.

Moreover, evaluating the effect of the GDPR (General Data Protection Regulations - making the user aware of third-party elements on websites in the European Union), would be an interesting future research focus. Here, one should evaluate if all third-party elements are presented to the user and can be deactivated. Further, an analysis of the user behavior would be interesting, how many users accept the GDPR terms without knowing the consequences. We also see a research focus on the effects of newly introduces tracking-protection methods in the Firefox and Safari browser.

While our solution is resistant to various tracking techniques, we have no means to prevent tracking via IP addresses. This should be solved by using an anonymity network such as Tor or by using proxy servers. Another opportunity for future work is the continuous improvement of tracking countermeasures and the integration of our solutions in all common browsers. Many of the improvements we have mentioned can probably be implemented using the add-on API of Firefox or Chrome and similar APIs of other browsers. Also, there is a need to develop a proper mobile browser protection, since the API, access rights as well as user interfaces are different to the one we used in the Chromium browser.

Bibliography

[1] M. Kosinski, D. Stillwell, and T. Graepel, "Private traits and attributes are predictable from digital records of human behavior," *Proceedings of the National Academy of Sciences*, vol. 110, no. 15, pp. 5802–5805, 2013.

[2] J. Angwin and J. Valentino-DeVries, "Race Is On to 'Fingerprint' Phones, PCs." http://www.wsj.com/articles/SB10001424052748704679204575646704100959546, 2010.

[3] P. Papadopoulos, N. Kourtellis, and E. P. Markatos, "Cookie synchronization: Everything you always wanted to know but were afraid to ask," *CoRR*, vol. abs/1805.10505, 2018.

[4] C. Cadwalladr and E. Graham-Harrison, "Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach." https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election, 2018.

[5] A. Chaabane, G. Acs, and M. A. Kaafar, "You Are What You Like! Information Leakage Through Users' Interests," in *NDSS Symposium 2012 - 19th Annual Network and Distributed System Security Symposium*, (San Diego, United States), pp. 1–14, Feb. 2012.

[6] J. Su, A. Shukla, S. Goel, and A. Narayanan, "De-anonymizing web browsing data with social networks," in *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, (Republic and Canton of Geneva, Switzerland), pp. 1261–1269, International World Wide Web Conferences Steering Committee, 2017.

[7] A. D. Miyazaki and A. Fernandez, "Consumer perceptions of privacy and security risks for online shopping," *Journal of Consumer Affairs*, vol. 35, no. 1, pp. 27–44, 2001.

[8] P. Laperdrix, W. Rudametkin, and B. Baudry, "Beauty and the Beast: Diverting modern web browsers to build unique browser fingerprints," in *37th IEEE Symposium on Security and Privacy (S&P 2016)*, (San Jose, United States), 5 2016.

[9] J. Brookman, P. Rouge, A. Alva, and C. Yeung, "Cross-device tracking: Measurement and disclosures," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 2, pp. 133–148, 2017.

[10] S. E. Oh, S. Li, and N. Hopper, "Fingerprinting keywords in search queries over tor," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 4, pp. 251–270, 2017.

[11] M. S. Siddiqui, "Evercookies: Extremely persistent cookies," *IJCSIS*, 2011.

[12] J. R. Mayer and J. C. Mitchell, "Third-Party Web Tracking: Policy and Technology," *IEEE Symposium on Security and Privacy*, 2012.

[13] F. Roesner, T. Kohno, and D. Wetherall, "Detecting and Defending Against Third-Party Tracking on the Web," in *Usenix NSDI*, 2012.

[14] A. Narayanan and V. Shmatikov, "How To Break Anonymity of the Netflix Prize Dataset," *CoRR*, 2006.

[15] J. Angwin and T. Mc Ginty, "Sites feed personal details to new tracking industry." The Wall Street Journal, http://online.wsj.com/article/SB10001424052748703977004575393173432219064.html, July 30, 2010.

[16] K. Lobosco, "Facebook friends could change your credit score." http://money.cnn.com/2013/08/26/technology/social/facebook-credit-score/index.html, 2013.

[17] The Economist, "Marketing information offers insurers another way to analyse risk." http://www.economist.com/node/21556263, 2012.

[18] S. Gittelman, V. Lange, C. A. Gotway Crawford, C. Okoro, E. Lieb, S. Dhingra, and E. Trimarchi, "A new source of data for public health surveillance: Facebook likes," *Journal of Medical Internet Research*, 2015.

[19] T. Vissers, N. Nikiforakis, N. Bielova, and W. Joosen, "Crying Wolf? On the Price Discrimination of Online Airline Tickets," in *HotPETs)*, 2014.

[20] J. Mikians, L. Gyarmati, V. Erramilli, and N. Laoutaris, "Detecting price and search discrimination on the internet," in *HotNets*, 2012.

[21] C. Duhigg, "How Companies Learn Your Secrets." http://www.nytimes.com/2012/02/19/magazine/shopping-habits.html?pagewanted=all&_r=0, 2012. Accessed on 2013-10-25.

[22] M. Schneider, M. Enzmann, and M. Stopczynski, "Web-tracking-report 2014," Tech. Rep. SIT-TR-2014-01, Fraunhofer-Institut für Sichere Informationstechnologie, Feb. 2014.

[23] M. Stopczynski and M. Zugelder, "Reducing user tracking through automatic web site state isolations," in *Information Security: 17th International Conference, ISC 2014, Hong Kong, China, October 12-14, 2014.*, pp. 309–327, Springer International Publishing, 2014.

[24] B. Krishnamurthy, F. Park, and C. E. Wills, "Privacy Diffusion on the Web : A Longitudinal Perspective," in *WWW*, pp. 541–550, ACM, 2009.

[25] G. Merzdovnik, M. Huber, D. Buhov, N. Nikiforakis, S. Neuner, M. Schmiedecker, and E. Weippl, "Block me if you can: A large-scale study of tracker-blocking tools," in *2017 IEEE European Symposium on Security and Privacy*, pp. 319–333, 4 2017.

[26] S. Baviskar and P. S. Thilagam, "Protection of Web User's Privacy by Securing Browser from Web Privacy Attacks," *IJCTA*, 2011.

[27] A. Karaj, S. Macbeth, R. Berson, and J. M. Pujol, "Whotracks.me: Monitoring the online tracking landscape at scale," *CoRR*, vol. abs/1804.08959, 2018.

[28] T. Li, H. Hang, M. Faloutsos, and P. Efstathopoulos, "Trackadvisor: Taking back browsing privacy from third-party trackers," in *Passive and Active Measurement - 16th International Conference, PAM 2015, New York, NY, USA, March 19-20, 2015, Proceedings*, pp. 277–289, 2015.

[29] S. Englehardt and A. Narayanan, "Online tracking: A 1-million-site measurement and analysis," in *Proceedings of ACM CCS 2016*, 2016.

[30] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless Monster: Exploring the Ecosystem of Web-based Device Fingerprinting," *IEEE Symposium on Security and Privacy*, 2013.

[31] B. Liu, A. Sheth, U. Weinsberg, J. Chandrashekar, and R. Govindan, "Adreveal: Improving transparency into online targeted advertising," in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, HotNets-XII, (New York, NY, USA), pp. 12:1–12:7, ACM, 2013.

[32] nugg.ad AG, "Predictive Behavioral Targeting." https://www.nugg.ad/en/smart-audience-platform/audience-toolbox.html, 2016. Accessed on 2016-02-19.

[33] A. Lambrecht and C. Tucker, "When does retargeting work? information specificity in online advertising," *Journal of Marketing Research*, vol. 50, no. 5, pp. 561–576, 2013.

[34] J. M. Carrascosa, J. Mikians, R. Cuevas, V. Erramilli, and N. Laoutaris, "I always feel like somebody's watching me: Measuring online behavioural advertising," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '15, (New York, NY, USA), pp. 13:1–13:13, ACM, 2015.

[35] C. Castelluccia, M.-A. Kaafar, and M.-D. Tran, "Betrayed by your ads!," in *Privacy Enhancing Technologies* (S. Fischer-Hübner and M. Wright, eds.), (Berlin, Heidelberg), pp. 1–17, Springer Berlin Heidelberg, 2012.

[36] M. Barbaro and T. Zeller, Jr., "A Face Is Exposed for AOL Searcher No. 4417749." http://www.nytimes.com/2006/08/09/technology/09aol.html?ex=1312776000&en=f6f61949c6da4d38&ei=5090, 2006. Accessed on 2013-10-25.

[37] P. Ohm, "Broken Promises of Privacy: Responding to the Surprising Failure of Anonymization," *UCLA Law Review*, 2009.

[38] P. G. Leon, B. Ur, R. Balebako, L. F. Cranor, R. Shay, and Y. Wang, "Why Johnny Can't Opt Out: A Usability Evaluation of Tools to Limit Online Behavioral Advertising," *CHI*, 2012.

[39] R. N. Timothy Libert, Lucas Graves, "Changes in third-party content on european news websites after gdpr," *Reuters Institute*, 2018.

[40] X. Hu and N. Sastry, "Characterising third party cookie usage in the EU after GDPR," *CoRR*, vol. abs/1905.01267, 2019.

[41] J. Sørensen and S. Kosta, "Before and after gdpr: The changes in third party presence at public and private european websites," in *The World Wide Web Conference*, WWW '19, (New York, NY, USA), pp. 1590–1600, ACM, 2019.

[42] B. Schneier, "The Internet is a surveillance state." http://edition.cnn.com/2013/03/16/opinion/schneier-internet-surveillance, 2013. Accessed on 2013-10-25.

[43] E. Steel and G. A. Fowler, "Facebook in Privacy Breach." http://online.wsj.com/article/SB10001424052702304772804575558484075236968.html, 2010. Accessed on 2013-10-25.

[44] C. Scientist and T. Italia, "Flash Cookies and Privacy II: Now with HTML5 and ETag Respawning," *World Wide Web Internet And Web Information Systems*, 2009.

[45] M. Mulazzani and P. Reschl, "Fast and reliable browser identification with javascript engine fingerprinting," *W2SP*, 2013.

[46] P. Eckersley, "How unique is your web browser?," *PETs*, 2010.

[47] A. Vastel, P. Laperdrix, W. Rudametkin, and R. Rouvoy, "Fp-stalker: Tracking browser fingerprint evolutions," in *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 728–741, 5 2018.

[48] N. Anderson, "Firm uses typing cadence to finger unauthorized users." http://arstechnica.com/tech-policy/2010/02/firm-uses-typing-cadence-to-finger-unauthorized-users, 2010. Accessed on 2013-10-30.

[49] K. Mowery and H. Shacham, "Pixel Perfect: Fingerprinting Canvas in HTML5," in *W2SP*, IEEE Computer Society, 2012.

[50] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel, "FPDetective: Dusting the web for fingerprinters," in *CCS*, 2013.

[51] K. Boda, Á. M. Földes, G. G. Gulyás, and S. Imre, "User tracking on the web via cross-browser fingerprinting," in *Information Security Technology for Applications*, pp. 31–46, Springer, 2012.

[52] K. Mowery and D. Bogenreif, "Fingerprinting information in JavaScript implementations," *W2SP*, 2011.

[53] T. Bujlow, V. Carela-Español, J. Solé-Pareta, and P. Barlet-Ros, "Web tracking: Mechanisms, implications, and defenses," *CoRR*, 2015.

[54] S. Li, M. Imani, and N. Hopper, "Measuring Information Leakage in Website Fingerprinting Attacks and Defenses," in *Proceedings of ACM CCS 2018*, 2018.

[55] T. Bujlow, V. Carela-Español, J. Solé-Pareta, and P. Barlet-Ros, "A survey on web tracking: Mechanisms, implications, and defenses," *Proceedings of the IEEE*, vol. 105, pp. 1476–1510, 8 2017.

[56] J. Mayer, "Tracking the Trackers: Microsoft Advertising." `http://cyberlaw.stanford.edu/node/6715`, 2011. Online.

[57] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The web never forgets: Persistent tracking mechanisms in the wild," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, (New York, NY, USA), pp. 674–689, ACM, 2014.

[58] A. Mathur, J. Vitak, A. Narayanan, and M. Chetty, "Characterizing the use of browser-based blocking extensions to prevent online tracking," in *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, (Baltimore, MD), pp. 103–116, USENIX Association, 2018.

[59] S. Murdoch, M. Perry, and E. Clark, "Tor: Cross-origin fingerprinting unlinkabilit." `https://www.torproject.org/projects/torbrowser/design/#fingerprinting-linkability`, 2014. Accessed on 2014-11-30.

[60] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*, WPES '11, (New York, NY, USA), pp. 103–114, ACM, 2011.

[61] Eyeo GmbH, "Allowing acceptable ads in Adblock Plus." `https://adblockplus.org/en/acceptable-ads`, 2014. Accessed on 2014-08-13.

[62] K. Garimella, O. Kostakis, and M. Mathioudakis, "Ad-blocking: A study on performance, privacy and counter-measures," in *Proceedings of the 2017 ACM on Web Science Conference*, WebSci '17, (New York, NY, USA), pp. 259–262, ACM, 2017.

[63] B. Shiller, J. Waldfogel, and J. Ryan, "The effect of ad blocking on website traffic and quality," *The RAND Journal of Economics*, vol. 49, no. 1, pp. 43–63, 2018.

[64] M. Malloy, M. McNamara, A. Cahn, and P. Barford, "Ad blockers: Global prevalence and impact," in *Proceedings of the 2016 Internet Measurement Conference*, IMC '16, (New York, NY, USA), pp. 119–125, ACM, 2016.

[65] R. Nithyanand, S. Khattak, M. Javed, N. Vallina-Rodriguez, M. Falahrastegar, J. E. Powles, E. D. Cristofaro, H. Haddadi, and S. J. Murdoch, "Adblocking and counter blocking: A slice of the arms race," in *6th USENIX Workshop on Free and Open Communications on the Internet (FOCI 16)*, (Austin, TX), USENIX Association, 2016.

[66] U. Iqbal, Z. Shafiq, and Z. Qian, "The ad wars: Retrospective measurement and analysis of anti-adblock filter lists," in *Proceedings of the 2017 Internet Measurement Conference*, IMC '17, (New York, NY, USA), pp. 171–183, ACM, 2017.

[67] M. H. Mughees, Z. Qian, and Z. Shafiq, "Detecting anti ad-blockers in the wild," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 3, pp. 130–146, 2017.

[68] M. Geuss, "Over 300 businesses now whitelisted on AdBlock Plus, 10play." https://arstechnica.com/information-technology/2015/02/over-300-businesses-now-whitelisted-on-adblock-plus-10-pay-to-play/, 2015.

[69] R. Bilton, "Ghostery: A Web tracking blocker that actually helps the ad industry." http://venturebeat.com/2012/07/31/ghostery-a-web-tracking-blocker-that-actually-helps-the-ad-industry, 2012. Accessed on 2014-08-13.

[70] M. Koop, E. Tews, and S. Katzenbeisser, "In-depth evaluation of redirect tracking and link usage," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 4, pp. 394 – 413, 2020.

[71] P. Baumann, S. Katzenbeisser, M. Stopczynski, and E. Tews, "Disguised chromium browser: Robust browser, flash and canvas fingerprinting protection," in *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, WPES '16, (New York, NY, USA), pp. 37–46, ACM, 2016.

[72] C. Hoofnagle, A. Soltani, N. Good, D. Wambach, and M. Ayenson, "Behavioral advertising: The offer you cannot refuse," *Harvard Law & Policy Review*, vol. 6, no. 2, pp. 273–296, 2012.

[73] J. B. Kristen Purcell and L. Rainie, "Search engine use." http://pewinternet.org/Reports/2012/Search-Engine-Use-2012.aspx, 2012.

[74] TRUSTe and Harris Interactive, "Privacy and online behavioral advertising." https://www.eff.org/files/truste-2011-consumer-behavioral-advertising-survey-results.pdf, 2012.

[75] P. Stone, "Pixel Perfect Timing Attacks with HTML5." White Paper, 2013. http://contextis.co.uk/files/Browser_Timing_Attacks.pdf.

[76] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, (New York, NY, USA), pp. 605–616, ACM, 2012.

[77] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning," in *Proceedings of ACM CCS 2018*, 2018.

[78] D. Agrawal and C. C. Aggarwal, "On the design and quantification of privacy preserving data mining algorithms," PODS '01, 2001.

[79] J. Parra-Arnau, D. Rebollo-Monedero, and J. Forne, "Measuring the privacy of user profiles in personalized information systems," *Future Generation Computer Systems*, vol. 33, pp. 53 – 63, 2014. Special Section on Applications of Intelligent Data and Knowledge Processing Technologies.

[80] M. Helft and T. Vega, "Retargeting ads follow surfers to other sites." New York Times, http://www.nytimes.com/2010/08/30/technology/30adstalk.html?_r=0, August 29, 2010.

[81] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas, "Adnostic: Privacy preserving targeted advertising," in *NDSS*, 2010.

[82] S. Ellis, "The future of retargeting, remarketing and remessaging." Marketing Land, http://marketingland.com/the-future-of-retargeting-remarketing-and-remessaging-7643, August 29, 2012.

[83] A. Goldfarb, "What is different about online advertising?," *Review of Industrial Organization*, vol. 43, no. 1-2, pp. 1–15, 2013.

[84] M. Falahrastegar, H. Haddadi, S. Uhlig, and R. Mortier, "Tracking personal identifiers across the web," in *Passive and Active Measurement* (T. Karagiannis and X. Dimitropoulos, eds.), (Cham), pp. 30–41, Springer International Publishing, 2016.

[85] S. Ward, "Web tracking has become a privacy time bomb." https://usatoday30.usatoday.com/tech/news/2011-08-03-internet-tracking-mobile-privacy_n.htm, 2011.

[86] eMarketer, "eMarketer Releases New Global Media Ad Spending Estimates." emarketer.com/content/emarketer-total-media-ad-spending-worldwide-will-rise-7-4-in-2018, 2018.

[87] campaign, "Group M unveils upbeat global adspend forecast for 2018." https://www.campaignlive.co.uk/article/group-m-unveils-upbeat-global-adspend-forecast-2018/1451805, 2017.

[88] PWC, "Global entertainment and media outlook." http://www.pwc.com/gx/en/industries/entertainment-media/outlook/segment-insights/internet-advertising.html, 2015.

[89] eMarketer, "Google Will Take 55% of Search Ad Dollars Globally in 2015." emarketer.com/Article/Google-Will-Take-55-of-Search-Ad-Dollars-Globally-2015/1012294, 2015.

[90] A. Nanji, "Global Ad Spend Forecast by Medium and Region." http://www.marketingprofs.com/charts/2015/27999/2015-global-ad-spend-forecast-by-medium-and-region, 2015.

[91] D. Evans, "The online advertising industry: Economics, evolution, and privacy," *Journal of Economic Perspectives*, vol. 23, no. 3, pp. 37–60, 2009.

[92] S. Guha, B. Cheng, and P. Francis, "Challenges in measuring online advertising systems," in *10th ACM SIGCOMM Conference on Internet Measurement (IMC 2010), Proceedings*, pp. 81–87, 2010.

[93] J. Harper, "It's modern trade: Web users get as much as they give." The Wall Street Journal, http://online.wsj.com/article/SB10001424052748703748904575411530096840958.html#, August 6, 2010.

[94] A. Goldfarb and C. Tucker, "Online advertising, behavioral targeting, and privacy," *Communications of the ACM*, vol. 54, no. 5, pp. 25–27, 2011.

[95] J. Yan, N. Liu, G. Wang, W. Zhang, Y. Jiang, and Z. Chen, "How much can behavioral targeting help online advertising?," in *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, (New York, NY, USA), pp. 261–270, ACM, 2009.

[96] R. Carroll, "The alchemy of behavioral targeting." http://www.destinationcrm.com/Articles/PrintArticle.aspx?ArticleID=81719, April 13, 2012.

[97] J. Angwin, "The web's new gold mine: Your secrets." The Wall Street Journal, http://online.wsj.com/article/SB10001424052748703940904575395073512989404.html, July 30, 2010.

[98] hootsuite, "Social Media Advertising Stats that Matter to Marketers in 2018." https://blog.hootsuite.com/social-media-advertising-stats/, 2017.

[99] D. Liberto, "Facebook, Google Digital Ad Market Share Drops as Amazon Climbs." https://www.investopedia.com/news/facebook-google-digital-ad-market-share-drops-amazon-climbs/, 2018.

[100] Spiegel Online, "Handel mit vertraulichen Daten – Millionen deutsche Patienten und Ärzte werden ausgespäht." http://www.spiegel.de/netzwelt/netzpolitik/patienten-apotheken-verkaufen-vertrauliche-daten-a-917118.html,Accessed on 2013-08-18, 2013.

[101] S. Yuan, A. Z. Abidin, M. Sloan, and J. Wang, "Internet advertising: An interplay among advertisers, online publishers, ad exchanges and web users," *CoRR*, vol. abs/1206.1754, 2012.

[102] B. Lincoln, "Motorola Is Listening." http://www.beneaththewaves.net/Projects/Motorola_Is_Listening.html, 2013. Accessed on 2013-10-25.

[103] R. Behar, "Never Heard Of Acxiom? Chances Are It's Heard Of You. How a little-known Little Rock company—the world's largest processor of consumer data—found itself at the center of a very big national security debate.." http://money.cnn.com/magazines/fortune/fortune_archive/2004/02/23/362182/index.htm, 2004. Accessed on 2013-10-25.

[104] A. Chaabane, M. Kaafar, and R. Boreli, "Big friend is watching you: analyzing online social networks tracking capabilities," in *WOSN*, pp. 7–12, 2012.

[105] J. Constine, "Facebook Lets Businesses Plug In CRM Email Addresses To Target Customers With Hyper-Relevant Ads." http://techcrunch.com/2012/09/20/facebook-crm-ads/, 2012. Accessed on 2013-10-30.

[106] C. Tucker, "Social networks, personalized advertising, and privacy controls." The Tenth Workshop on Economics of Information Security (WEIS 2011), http://weis2011.econinfosec.org/papers/Social%20Networks,%20Personalized%20Advertising,%20and%20Privacy%20Cont.pdf, June 2011.

[107] L. Zeltser, "Malvertising: The Use of Malicious Ads to Install Malware." http://infosecisland.com/blogview/14371-Malvertising-The-Use-of-Malicious-Ads-to-Install-Malware.html, 2011. Accessed on 2013-10-30.

[108] A. Sood and R. Enbody, "Malvertising - exploiting web advertising," *Computer Fraud and Security*, vol. 2011, no. 4, pp. 11 – 16, 2011.

[109] X. Xing, W. Meng, B. Lee, U. Weinsberg, A. Sheth, R. Perdisci, and W. Lee, "Understanding malvertising through ad-injecting browser extensions," in *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, (Republic and Canton of Geneva, Switzerland), pp. 1286–1295, International World Wide Web Conferences Steering Committee, 2015.

[110] E. Kosta, C. Kalloniatis, L. Mitrou, and E. Kavakli, "Search engines: Gateway to a new panopticon," in *Trust, Privacy and Security in Digital Business, 6th International Conference (TrustBus 2009), Proceedings*, vol. 5695 of *Lecture Notes on Computer Science*, pp. 11–21, Springer Verlag, 2009.

[111] L. Kaufman, "How private is the internet?," *IEEE Security & Privacy*, vol. 9, no. 1, pp. 73–75, 2011.

[112] A. Hannak, P. Sapieżyński, A. M. Kakhki, B. Krishnamurthy, D. Lazer, A. Mislove, and C. Wilson, "Measuring Personalization of Web Search," in *WWW*, 2013.

[113] A. Gunawardana and C. Meek, "Aggregators and Contextual Effects in Search Ad Markets," in *WWW Workshop on Targeting and Ranking for Online Advertising*, 4 2008.

[114] J. Vascellaro, "Google agonizes on privacy as ad world vaults ahead." The Wall Street Journal, http://online.wsj.com/article/SB10001424052748703309704575413553851854026.html, August 9, 2010.

[115] J. Healey, "Privacy advocates attack gmail - again - for email scanning." Los Angeles Times, August 15, 2013, http://www.latimes.com/news/opinion/opinion-la/la-ol-google-gmail-privacy-reasonable-expectation-20130814,0,2662122.story, 2013.

[116] B. Krishnamurthy and K. Naryshkin, "Privacy leakage vs. Protection measures: the growing disconnect," *Web 2.0 Security and Privacy Workshop*, 2011.

[117] A. Narayanan, "There is no such thing as anonymous online tracking." http://cyberlaw.stanford.edu/blog/2011/07/there-no-such-thing-anonymous-online-tracking, 2011.

[118] A. Narayanan, "Yet Another Identity Stealing Bug. Will Creeping Normalcy be the Result?." http://33bits.org/2010/06/01/yet-another-identity-stealing-bug-will-creeping-normalcy-be-the-result/, 2010.

[119] A. Narayanan, "How Google Docs Leaks Your Identity." http://33bits.org/2010/02/22/google-docs-leaks-identity/, 2010.

[120] A. Narayanan, "Ubercookies Part 2: History Stealing meets the Social Web." http://33bits.org/2010/02/19/ubercookies-history-stealing-social-web/, 2010.

[121] A. Narayanan, "Cookies, Supercookies and Ubercookies: Stealing the Identity of Web Visitors." http://33bits.org/2010/02/18/cookies-supercookies-and-ubercookies-stealing-the-identity-of-web-visitors/, 2010.

[122] A. Narayanan, "Facebook's Instant Personalization: An Analysis of Fundamental Privacy Flaws." http://33bits.org/2010/09/28/instant-personalization-privacy-flaws/, 2010.

[123] L.-S. Huang and C. Jackson, "Clickjacking Attacks Unresolved." http://nhonhangheo.blogspot.ca/2015/05/clickjacking-attacks-unresolved.html, 2015.

[124] L. Sweeney, "Weaving technology and policy together to maintain confidentiality," *The Journal of Law, Medicine and Ethics*, vol. 25, no. 2-3, pp. 98–110, 1997.

[125] L. Sweeney, "Uniqueness of Simple Demographics in the U.S. Population," 2000.

[126] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *IEEE Symposium on Security and Privacy*, pp. 111–125, 2008.

[127] S. Vimercati and S. Foresti, *Encyclopedia of Cryptography and Security*, ch. Quasi-Identifier, pp. 1010–1011. Boston, MA: Springer US, 2011.

[128] A. Narayanan and V. Shmatikov, "De-anonymizing Social Networks," *IEEE Symposium on Security and Privacy*, 2009.

[129] G. Friedland, G. Maier, R. Sommer, and N. Weaver, "Sherlock holmes' evil twin: on the impact of global inference for online privacy," in *New security paradigms workshop (NSPW 2011), Proceedings*, pp. 105–114, 2011.

[130] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel, "A practical attack to de-anonymize social network users," in *IEEE Symposium on Security and Privacy*, pp. 223–238, 2010.

[131] G. Himmelein, "Bericht: Apotheken verkaufen ungenügend anonymisierte Patientendaten." http://www.heise.de/newsticker/meldung/Bericht-Apotheken-verkaufen-ungenuegend-anonymisierte-Patientendaten-1937568.html, 2013. Accessed on 2013-10-25.

[132] J. Valentino-Devries, J. Singer-Vine, and A. Soltani, "Websites Vary Prices, Deals Based on Users' Information." http://online.wsj.com/news/articles/SB10001424127887323777204578189391813881534, 2012. Accessed on 2013-11-25.

[133] D. Mattioli, "On Orbitz, Mac Users Steered to Pricier Hotels." http://online.wsj.com/news/articles/SB10001424052702304458604577488822667325882, 2012. Accessed on 2013-10-25.

[134] A. Hannak, G. Soeller, D. Lazer, A. Mislove, and C. Wilson, "Measuring Price Discrimination and Steering on E-commerce Web Sites," in *Proceedings of the 14th ACM/USENIX Internet Measurement Conference (IMC'14)*, (Vancouver, Canada), 11 2014.

[135] Spiegel Online, "German Agency to Mine Facebook to Assess Creditworthiness." http://www.spiegel.de/international/germany/german-credit-agency-plans-to-analyze-individual-facebook-pages-a-837539.html, 2012.

[136] J. Barnes, "Big data bring risks and benefits to insurance customers." http://www.ft.com/cms/s/0/21e289c4-97ef-11e3-8dc3-00144feab7de.html#axzz41oCbtf9J, 2014.

[137] L. Scism and M. Maremont, "Insurers Test Data Profiles to Identify Risky Clients." http://www.wsj.com/news/articles/SB10001424052748704648604575620750998072986, 2010.

[138] B. Gellman and L. Poitras, "U.S., British intelligence mining data from nine U.S. Internet companies in broad secret program." http://www.washingtonpost.com/investigations/us-intelligence-mining-data-from-nine-us-internet-companies-in-broad-secret-program/2013/06/06/3a0c0da8-cebf-11e2-8845-d970ccb04497_print.html, 2013. Accessed on 2013-10-30.

[139] P. Bump, "The UK Tempora Program Captures Vast Amounts of Data - and Shares with NSA." http://www.theatlanticwire.com/national/2013/06/uk-tempora-program/66490/, 2013. Accessed on 2013-10-30.

[140] C. Timberg, "The NSA slide you haven't seen." http://www.washingtonpost.com/business/economy/the-nsa-slide-you-havent-seen/2013/07/10/32801426-e8e6-11e2-aa9f-c03a72e2d342_print.html, 2013. Accessed on 2013-10-30.

[141] D. McCullagh, "Feds put heat on Web firms for master encryption keys." http://news.cnet.com/8301-13578_3-57595202-38/feds-put-heat-on-web-firms-for-master-encryption-keys/, 2013. Accessed on 2013-10-30.

[142] Google Inc., "Transparency Report - Requests for user information." https://www.google.com/transparencyreport/userdatarequests/, 2015.

[143] G. Greenwald and S. Ackerman, "How the NSA is still harvesting your online data ." http://www.theguardian.com/world/2013/jun/27/nsa-online-metadata-collection, 2013.

[144] A. Soltani, A. Peterson, and B. Gellman, "NSA uses Google cookies to pinpoint targets for hacking." https://www.washingtonpost.com/news/the-switch/wp/2013/12/10/nsa-uses-google-cookies-to-pinpoint-targets-for-hacking, 2013.

[145] The Guardian, "Tor Stinks." http://www.theguardian.com/world/interactive/2013/oct/04/tor-stinks-nsa-presentation-document, 2013.

[146] H. Beales, "The Value of Behavioral Targeting," *Network Advertising Initiative*, 2010.

[147] N. Schmücker, "Web tracking." SNET2 Seminar Paper. Berlin University of Technology, 2011.

[148] K. McKinley, "Cleaning Up After Cookies." https://www.isecpartners.com/media/11976/isec_cleaning_up_after_cookies.pdf, 2008.

[149] T. Frank, "Javascript session variables without cookies." http://www.thomasfrank.se/sessionvars.html, 2008.

[150] M. Tran, X. Dong, Z. Liang, and X. Jiang, "Tracking the trackers: fast and scalable dynamic analysis of web content for privacy violations," *ACNS*, 2012.

[151] J. Bau, J. Mayer, H. Paskov, and J. Mitchell, "A Promising Direction for Web Tracking Countermeasures," in *W2SP*, 2013.

[152] Comscore, Inc., "The Impact of Cookie Deletion on Site-Server and Ad- Server Metrics in Australia." http://www.comscore.com/content/download/7251/125689/file/Impact+of+Cookie+Deletion+Australia_January+2011.pdf, 2011.

[153] A. Lerner, A. K. Simpson, T. Kohno, and F. Roesner, "Internet jones and the raiders of the lost trackers: An archaeological study of web tracking from 1996 to 2016," in *25th USENIX Security Symposium (USENIX Security 16)*, (Austin, TX), USENIX Association, 2016.

[154] G. Fleischer, "Implementing web tracking." https://media.blackhat.com/bh-us-12/Briefings/Fleischer/BH_US_12_Fleischer_Implementing_Web_Tracking_gfleischer_WP.pdf, 2012. Black Hat.

[155] S. Mittal, "User Privacy and the Evolution of Third-party Tracking Mechanisms on the World Wide Web," 2012.

[156] Oracle, "PersistenceService." https://docs.oracle.com/javase/7/docs/jre/api/javaws/jnlp/javax/jnlp/PersistenceService.html, 2014.

[157] G. Aggarwal, E. Bursztein, C. Jackson, and D. Boneh, "An analysis of private browsing modes in modern browsers," in *Usenix Security Symposium*, 2010.

[158] S. Kamkar, "evercookie - virtually irrevocable persistent cookies." http://samy.pl/evercookie, 2011. Accessed on 2015-02-10.

[159] E. Felten and M. Schneider, "Timing attacks on Web privacy," in *CCS*, 2000.

[160] E. Bursztein, "Tracking users that block cookies with a HTTP redirect." http://elie.im/blog/security/tracking-users-that-block-cookies-with-a-http-redirect, 2011.

[161] J. Grossman, "Tracking users with Basic Auth." http://jeremiahgrossman.blogspot.com.es/2007/04/trackingusers-without-cookies.html, 2007.

[162] Tor Bug Tracker, "Disable TLS Session resumption and Session IDs." https://trac.torproject.org/projects/tor/ticket/4099, 2011.

[163] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail," in *2012 IEEE Symposium on Security and Privacy*, pp. 332–346, 5 2012.

[164] C. V. Wright, S. E. Coull, and F. Monrose, "Traffic morphing: An efficient defense against statistical traffic analysis," in *In Proceedings of the 16th Network and Distributed Security Symposium*, pp. 237–250, IEEE, 2009.

[165] M. Perry, "Experimental defense for website traffic fingerprinting." https://blog.torproject.org/blog/experimental-defense-website-traffic-fingerprinting, 2011.

[166] X. Luo, P. Zhou, E. W. W. Chan, W. Lee, R. K. C. Chang, and R. Perdisci, "Httpos: Sealing information leaks with browser-side obfuscation of encrypted flows," in *In Proc. Network and Distributed Systems Symposium (NDSS). The Internet Society*, 2011.

[167] T. Wang and I. Goldberg, "Improved website fingerprinting on tor," in *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, WPES '13, (New York, NY, USA), pp. 201–212, ACM, 2013.

[168] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A critical evaluation of website fingerprinting attacks," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, (New York, NY, USA), pp. 263–274, ACM, 2014.

[169] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proceedings of the 23rd USENIX Conference on Security Symposium*, SEC'14, (Berkeley, CA, USA), pp. 143–157, USENIX Association, 2014.

[170] X. Cai, R. Nithyanand, and R. Johnson, "Cs-buflo: A congestion sensitive website fingerprinting defense," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, WPES '14, (New York, NY, USA), pp. 121–130, ACM, 2014.

[171] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, "A systematic approach to developing and evaluating website fingerprinting defenses," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, (New York, NY, USA), pp. 227–238, ACM, 2014.

[172] R. Nithyanand, X. Cai, and R. Johnson, "Glove: A bespoke website fingerprinting defense," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, WPES '14, (New York, NY, USA), pp. 131–134, ACM, 2014.

[173] M. Perry, "A Critique of Website Traffic Fingerprinting Attacks." https://blog.torproject.org/blog/critique-website-traffic-fingerprinting-attacks, 2015.

[174] T. Wang and I. Goldberg, "On realistically attacking tor with website fingerprinting," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 4, pp. 21–36, 2016.

[175] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel, "Website fingerprinting at Internet scale," in *NDSS*, The Internet Society, 2016.

[176] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *25th USENIX Security Symposium (USENIX Security 16)*, (Austin, TX), pp. 1187–1203, USENIX Association, 2016.

[177] T. Wang and I. Goldberg, "Walkie-talkie: An efficient defense against passive website fingerprinting attacks," in *Proceedings of the 26th USENIX Conference on Security Symposium*, SEC'17, (Berkeley, CA, USA), pp. 1375–1390, USENIX Association, 2017.

[178] G. Cherubin, J. Hayes, and M. Juarez, "Website fingerprinting defenses at the application layer," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 2, pp. 186–203, 2017.

[179] N. Nikiforakis, W. Joosen, and B. Livshits, "Privaricator: Deceiving fingerprinters with little white lies," in *research.microsoft.com*, 2014.

[180] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, pp. 93–108, 4 2005.

[181] T. Yen, Y. Xie, F. Yu, R. Yu, and M. Abadi, "Host fingerprinting and tracking on the web: Privacy and security implications," 2012.

[182] L. Olejnik, G. Acar, C. Castelluccia, and C. Díaz, "The leaking battery: A privacy analysis of the HTML5 battery status API," *IACR Cryptology ePrint Archive*, vol. 2015.

[183] A. Kurtz, H. Gascon, T. Becker, K. Rieck, and F. C. Freiling, "Fingerprinting mobile devices using personalized configurations," *PoPETs*, vol. 2016, pp. 4–19, 2016.

[184] S. Seneviratne, A. Seneviratne, P. Mohapatra, and A. Mahanti, "Predicting user traits from a snapshot of apps installed on a smartphone," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 18, pp. 1–8, June 2014.

[185] J. R. Corripio, D. M. A. González, A. L. S. Orozco, L. J. G. Villalba, J. Hernandez-Castro, and S. J. Gibson, "Source smartphone identification using sensor pattern noise and wavelet transform," in *5th International Conference on Imaging for Crime Detection and Prevention (ICDP 2013)*, pp. 1–6, 12 2013.

[186] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh, "Mobile device identification via sensor fingerprinting," *CoRR*, vol. abs/1408.1416, 2014.

[187] A. Das, N. Borisov, and M. Caesar, "Do you hear what i hear?: Fingerprinting smart devices through embedded acoustic components," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, (New York, NY, USA), pp. 441–452, ACM, 2014.

[188] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi, "Accelprint: Imperfections of accelerometers make smartphones trackable," in *NDSS*, The Internet Society, 2014.

[189] Z. Zhou, W. Diao, X. Liu, and K. Zhang, "Acoustic fingerprinting revisited: Generate stable device id stealthily with inaudible sound," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, (New York, NY, USA), pp. 429–440, ACM, 2014.

[190] W. D. Yu, S. Nargundkar, and N. Tiruthani, "A phishing vulnerability analysis of web based systems," pp. 326–331, 7 2008.

[191] J. Chen and C. Guo, "Online detection and prevention of phishing attacks," in *2006 First International Conference on Communications and Networking in China*, pp. 1–7, 10 2006.

[192] K. Tian, K. Cooper, K. Zhang, and S. Liu, "Towards a new understanding of advice interference," in *Fourth International Conference on Secure Software Integration and Reliability Improvement, SSIRI 2010, Singapore, June 9-11, 2010*, pp. 180–189, 2010.

[193] S. Abu-Nimeh and S. Nair, "Circumventing security toolbars and phishing filters via rogue wireless access points," *Wireless Communications and Mobile Computing*, vol. 10, no. 8, pp. 1128–1139, 2010.

[194] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose, "All your iframes point to us," in *Proceedings of the 17th Conference on Security Symposium*, SS'08, (Berkeley, CA, USA), pp. 1–15, USENIX Association, 2008.

[195] J. Zhang, C. Seifert, J. W. Stokes, and W. Lee, "Arrow: Generating signatures to detect drive-by downloads," in *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, (New York, NY, USA), pp. 187–196, ACM, 2011.

[196] M. Cova, C. Kruegel, and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious javascript code," in *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, (New York, NY, USA), pp. 281–290, ACM, 2010.

[197] J. Mayer, "How Verizon's Advertising Header Works | Web Policy." http://webpolicy.org/2014/10/24/how-verizons-advertising-header-works, 2014.

[198] D. Arp, E. Quiring, C. Wressnegger, and K. Rieck, "Privacy threats through ultrasonic side channels on mobile devices," in *2017 IEEE European Symposium on Security and Privacy*, pp. 35–47, 4 2017.

[199] D. Goodin, "Beware of ads that use inaudible sound to link your phone, TV, tablet, and PC." https://arstechnica.com/tech-policy/2015/11/beware-of-ads-that-use-inaudible-sound-to-link-your-phone-tv-tablet-and-pc/, 2015.

[200] A. Soltani, S. Canty, Q. Mayo, L. Thomas, and C. J. Hoofnagle, "Flash cookies and privacy," in *AAAI*, 2010.

[201] S. Murdoch, M. Perry, and E. Clark, "Identifier Unlinkability Defenses in the Tor Browser." https://www.torproject.org/projects/torbrowser/design/, 2014.

[202] J. Turow, J. King, C. Hoofnagle, A. Bleakley, and M. Hennessy, "Americans reject tailored advertising and three activities that enable it." Social Science Research Network, SSRN Working Paper Series, http://ssrn.com/abstract=1478214, 2009.

[203] E. Pujol, O. Hohlfeld, and A. Feldmann, "Annoyed users: Ads and ad-block usage in the wild," in *Proceedings of the 2015 Internet Measurement Conference*, IMC '15, (New York, NY, USA), pp. 93–106, ACM, 2015.

[204] Pagefair, "Ad Blocking Report." https://blog.pagefair.com/2015/ad-blocking-report, 2015.

[205] PageFair, "The state of the blocked web." https://pagefair.com/downloads/2017/01/PageFair-2017-Adblock-Report.pdf, 2017.

[206] A. Zeigler, A. Bateman, and E. Graff, "Web Tracking Protection." https://www.w3.org/Submission/2011/SUBM-web-tracking-protection-20110224/#list-format/, 2016.

[207] Mozilla Foundation, "Tracking Protection." https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Privacy/Tracking_Protection, 2011.

[208] Mozilla Foundation, "Private Browsing." https://blog.mozilla.org/press-de/2015/12/16/mehr-kontrolle-uber-private-daten-im-private-browsing-modus-von-firefox/, 2015.

[209] Mozilla Foundation, "Wie wir den Tracking-Schutz in Firefox noch besser machen möchten." hhttps://blog.mozilla.org/press-de/2018/08/30/wie-wir-den-tracking-schutz-in-firefox-noch-besser-machen-moechten/, 2018.

[210] Opera Software, "Introducing native ad-blocking feature for faster browsing." https://blogs.opera.com/desktop/2016/03/native-ad-blocking-feature-opera-for-computers/, 2016.

[211] A. Morris, "Apple's iOS 9 Creates a Digital Marketing Crisis in the Wake of Ad Blocker Success." https://www.allbusiness.com/ios-9-ad-blockers-100836-1.html, 2015.

[212] Webkit, "Intelligent Tracking Prevention." https://webkit.org/blog/7675/intelligent-tracking-prevention/, 2017.

[213] Webkit, "Intelligent Tracking Prevention 1.1." https://webkit.org/blog/8142/intelligent-tracking-prevention-1-1/, 2018.

[214] Webkit, "Intelligent Tracking Prevention 2.0." https://webkit.org/blog/8311/intelligent-tracking-prevention-2-0/, 2018.

[215] Network Advertising Initiative, "Opt Out of Interest-Based Advertising." http://www.networkadvertising.org/choices/, 2016.

[216] W3C, "Web Tracking and User Privacy Workshop." http://www.w3.org/2011/04/29-w3cdnt-minutes.html, 2011.

[217] J. Mayer and A. Narayanan, "Do Not Track - Universal Web Tracking Opt Out." http://www.donottrack.us/, 2010.

[218] FTC, "FTC Staff Issues Privacy Report - Endorses Do Not Track to Facilitate Consumer Choice About Online Tracking." https://www.ftc.gov/news-events/press-releases/2010/12/ftc-staff-issues-privacy-report-offers-framework-consumers, 2010.

[219] J. Mayer, A. Narayanan, and S. Stamm, "Do not track: A universal third-party web tracking opt out." IETF Internet-Draft, http://tools.ietf.org/html/draft-mayer-do-not-track-00, March 7, 2011.

[220] R. Fielding and D. Singer, "Tracking preference expression (dnt)." W3C Working Draft, http://www.w3.org/TR/tracking-dnt, September 12, 2013.

[221] R. Reitman, "Mozilla Leads the Way on Do Not Track." https://www.eff.org/deeplinks/2011/01/mozilla-leads-the-way-on-do-not-track, 2011.

[222] N. Kroes, "Online privacy and online business: An update on do not track." European Commission, Press Releases Database, http://europa.eu/rapid/press-release_SPEECH-12-716_en.htm, October 11, 2012.

[223] Network Advertising Initiative, "EU legislation on cookies." http://ec.europa.eu/ipg/basics/legal/cookies/index_en.htm, 2009.

[224] J. Mazel, R. Garnier, and K. Fukuda, "A comparison of web privacy protection techniques," *CoRR*, vol. abs/1712.06850, 2017.

[225] R. J. Walls, E. D. Kilmer, N. Lageman, and P. D. McDaniel, "Measuring the impact and perception of acceptable advertisements," in *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, IMC '15, (New York, NY, USA), pp. 107–120, ACM, 2015.

[226] D. Jang, R. Jhala, S. Lerner, and H. Shacham, "An Empirical Study of Privacy-Violating Information Flows in JavaScript Web Applications," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, (New York, NY, USA), pp. 270–283, ACM, 2010. http://cseweb.ucsd.edu/~d1jang/papers/ccs10.pdf.

[227] M. Ikram, H. J. Asghar, M. A. Kâafar, A. Mahanti, and B. Krishnamurthy, "Towards seamless tracking-free web: Improved detection of trackers via one-class learning," *PoPETs*, vol. 2017, no. 1, pp. 79–99, 2017.

[228] G. Storey, D. Reisman, J. Mayer, and A. Narayanan, "The future of ad blocking: An analytical framework and new techniques," *CoRR*, vol. abs/1705.08568, 2017.

[229] Security-in-a-Box, "Remain anonymous and bypass censorship on the Internet." https://securityinabox.org/en/guide/anonymity-and-circumvention/, 2015.

[230] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," in *Usenix Security Symposium*, 2004.

[231] S. J. Murdoch and P. Zieliński, "Sampled Traffic Analysis by Internet-Exchange-level Adversaries," PETs, 2007.

[232] A. Acquisti, L. K. John, and G. Loewenstein, "What is privacy worth?," *The Journal of Legal Studies*, vol. 42, no. 2, pp. 249 – 274, 2013.

[233] Logicalis, "The age of digital enlightenment." https://www.uk.logicalis.com/globalassets/united-kingdom/microsites/real-time-generation/realtime-generation-2016-report.pdf, 2017. Accessed on 2018-10-25.

[234] K. C. Gina Pingitore, Vikram Rao and K. Dwivedi, "To share or not to share." https://www2.deloitte.com/content/dam/insights/us/articles/4020_To-share-or-not-to-share/DUP_To-share-or-not-to-share.pdf, 2017.

[235] J. Estrada-Jiménez, J. Parra-Arnau, A. Rodríguez-Hoyos, and J. Forné, "Online advertising: Analysis of privacy threats and protection approaches," *Computer Communications*, vol. 100, pp. 32 – 51, 2017.

[236] A. Juels, "Targeted advertising ... and privacy too," in *Proceedings of the 2001 Conference on Topics in Cryptology: The Cryptographer's Track at RSA*, CT-RSA 2001, (Berlin, Heidelberg), pp. 408–424, Springer-Verlag, 2001.

[237] C. Riederer, V. Erramilli, A. Chaintreau, B. Krishnamurthy, and P. Rodriguez, "For sale : Your data: By : You," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, (New York, NY, USA), pp. 13:1–13:6, ACM, 2011.

[238] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas, "Adnostic: Privacy preserving targeted advertising," in *NDSS*, 2010.

[239] S. Guha, B. Cheng, and P. Francis, "Privad: Practical privacy in online advertising," in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, (Berkeley, CA, USA), pp. 169–182, USENIX Association, 2011.

[240] M. Fredrikson and B. Livshits, "Repriv: Re-envisioning in-browser privacy," 2010.

[241] M. Backes, A. Kate, M. Maffei, and K. Pecina, "Obliviad: Provably secure and practical online behavioral advertising," in *2012 IEEE Symposium on Security and Privacy*, pp. 257–271, 5 2012.

[242] J. P. Achara, J. Parra-Arnau, and C. Castelluccia, "Mytrackingchoices: Pacifying the ad-block war by enforcing user privacy preferences," *CoRR*, vol. abs/1604.04495, 2016.

[243] J. Parra-Arnau, J. P. Achara, and C. Castelluccia, "Myadchoices: Bringing transparency and control to online advertising," *ACM Trans. Web*, vol. 11, pp. 7:1–7:47, Mar. 2017.

[244] C. Lu Wang, Y. J. Zhang, L. Richard Ye, and D. Nguyen, "Subscription to fee-based online services: What makes consumer pay for online content?," vol. 6, 01 2005.

[245] B. Software, "Basic attention token." https://basicattentiontoken.org/faq/BasicAttentionTokenWhitePaper-4.pdf, 2018.

[246] S. Hohenberger, S. Myers, R. Pass, and a. shelat, "Anonize: A large-scale anonymous survey system," in *2014 IEEE Symposium on Security and Privacy*, pp. 375–389, 5 2014.

[247] G. Keizer, "Brave browser begins controversial ad repeal-and-replace tests." https://www.computerworld.com/article/3284076/web-browsers/brave-browser-begins-controversial-ad-repeal-and-replace-tests.html, April 13, 2018.

[248] N. Lomas, "Shine signs first european carriers to its network-level ad blocking tech." https://techcrunch.com/2016/02/18/shine-bags-first-european-carrier-as-three-uk-deploys-network-level-ad-blocking/, 2015.

[249] bloomberg, "Inside the brotherhood of the ad blockers." https://www.bloomberg.com/news/features/2018-05-10/inside-the-brotherhood-of-pi-hole-ad-blockers, 2018.

[250] Beauftragte für den Datenschutz der EKD, ""pi-hole - ein erfahrungsbericht"." https://datenschutz.ekd.de/2018/04/12/pi-hole-ein-erfahrungsbericht/, 2018.

[251] K. Chellapilla and A. Maykov, "A taxonomy of javascript redirection spam," in *Proceedings of the 3rd International Workshop on Adversarial Information Retrieval on the Web*, AIRWeb '07, (New York, NY, USA), pp. 81–88, ACM, 2007.

[252] K. Bhargrava, B. Gsrc, D. Brewer, B. Gsrc, and B. Gsrc, "A Study of URL Redirection Indicating Spam," in *CEAS*, 2009.

[253] L. Li, X. Jin, S. J. Pan, and J. Sun, "Multi-domain active learning for text classification," in *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pp. 1086–1094, 2012.

[254] M. Gandhi, M. Jakobsson, and J. Ratkiewicz, "Badvertisements: Stealthy Click-Fraud with Unwitting Accessories," *Journal of Digital Forensic Practice*, vol. 1, pp. 131–142, 2006.

[255] C. E. Wills and D. C. Uzunoglu, "What ad blockers are (and are not) doing," in *2016 Fourth IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, pp. 72–77, 10 2016.

[256] G. Franken, T. V. Goethem, and W. Joosen, "Who left open the cookie jar? a comprehensive evaluation of third-party cookie policies," in *27th USENIX Security Symposium (USENIX Security 18)*, (Baltimore, MD), pp. 151–168, USENIX Association, 2018.

[257] V. Kalavri, J. Blackburn, M. Varvello, and K. Papagiannaki, "Like a Pack of Wolves: Community Structure of Web Trackers," in *Passive and Active Measurement*, pp. 42–54, Springer International Publishing, 2016.

[258] R. Gomer, E. M. Rodrigues, N. Milic-Frayling, and M. C. Schraefel, "Network analysis of third party tracking: User exposure to tracking cookies through search," in *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, vol. 1, pp. 549–556, 11 2013.

[259] M. A. Bashir and C. Wilson, "Diffusion of User Tracking Data in the Online Advertising Ecosystem," in *Proceedings on Privacy Enhancing Technologies (PETS 2018)*, (Barcelona, Spain), July 2018.

[260] S. Schelter and J. Kunegis, "On the ubiquity of web tracking: Insights from a billion-page web crawl," *J. Web Science*, vol. 4, pp. 53–66, 2018.

[261] P. Syverson and M. Traudt, "HSTS supports targeted surveillance," in *8th USENIX Workshop on Free and Open Communications on the Internet (FOCI 18)*, (Baltimore, MD), USENIX Association, Aug. 2018.

[262] B. Fulgham, "Protecting against hsts abuse." https://webkit.org/blog/8146/protecting-against-hsts-abuse/, 2018. Accessed on 2020-02-02.

[263] J. Wilander, "Preventing tracking prevention tracking." https://webkit.org/blog/9661/preventing-tracking-prevention-tracking/, 2019. Accessed on 2020-02-02.

[264] Mozilla, "Security/Anti tracking policy." https://wiki.mozilla.org/Security/Anti_tracking_policy, 2019. Accessed on 2020-02-02.

[265] C. Matte, N. Bielova, and C. Santos, "Do cookie banners respect my choice? measuring legal compliance of banners from iab europe's transparency and consent framework," 2019.

[266] C. Jackson, A. Bortz, D. Boneh, and J. C. Mitchell, "Protecting browser state from web privacy attacks," 2006.

[267] I. Fouad, N. Bielova, A. Legout, and N. Sarafijanovic-Djukic, "Missed by filter lists: Detecting unknown third-party trackers with invisible pixels," in *PETS 2020-20th Privacy Enhancing Technologies Symposium*, 2020.

[268] A. Clauset, C. R. Shalizi, and M. E. J. Newman, "Power-Law Distributions in Empirical Data," *SIAM Rev.*, 2009.

[269] J. Wilander, "Full Third-Party Cookie Blocking and More." https://webkit.org/blog/10218/full-third-party-cookie-blocking-and-more/, 2020.

[270] S. Englehardt, D. Reisman, C. Eubank, P. Zimmerman, J. Mayer, A. Narayanan, and E. W. Felten, "Cookies that give you away: The surveillance implications of web tracking," in *Proceedings of the 24th International Conference on World Wide Web*, pp. 289–299, 2015.

[271] P. E. Black, "Ratcliff/obershelp pattern recognition," *Dictionary of algorithms and data structures*, vol. 17, 2004.

[272] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008.

[273] J. Wilander, "Intelligent tracking prevention 2.0." https://webkit.org/blog/8311/intelligent-tracking-prevention-2-0/, June 2018.

[274] S. Murdoch, M. Perry, and E. Clark, "Tor: Cross-Origin Identifier Unlinkability." https://2019.www.torproject.org/projects/torbrowser/design/#identifier-linkability, 2018. Accessed on 2020-04-20.

[275] Mozilla, "Tor Uplift Project." https://wiki.mozilla.org/Security/Tor_Uplift, 2017. Accessed on 2020-04-20.

[276] Tor Project, "Tor at the Heart: Firefox." https://blog.torproject.org/tor-heart-firefox, 2016. Accessed on 2020-04-20.

[277] M. Brinkmann, "Mozilla adds Dynamic First Party Isolation option to Firefox 77." https://www.ghacks.net/2020/04/17/mozilla-adds-dynamic-first-party-isolation-option-to-firefox-77/, 2020. Accessed on 2020-04-30.

[278] M. Bostock, V. Ogievetsky, and J. Heer, "D3 data-driven documents," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, pp. 2301–2309, 2011.

[279] S. Englehardt, "Firefox 72 blocks third-party fingerprinting resources." https://blog.mozilla.org/security/2020/01/07/firefox-72-fingerprinting/, 2020.

[280] C. Reis, J. Dunagan, H. J. Wang, O. Dubrovsky, and S. Esmeir, "BrowserShield: Vulnerability-Driven Filtering of Dynamic HTML," *ACM Trans. Web*, 2007.

[281] G. Tavares, "Thoughts on asm.js vs PNaCl." http://games.greggman.com/game/thoughts-on-asm-js-vs-pnacl/, 2013. Accessed on 2013-10-30.

[282] Z. Weinberg, E. Chen, and C. Jackson, "I Still Know What You Visited Last Summer: Leaking Browsing History Via User Interaction and Side Channel Attacks," in *IEEE Symposium on Security and Privacy*, 2011.

[283] E. Y. Chen, J. Bau, C. Reis, A. Barth, and C. Jackson, "App Isolation: Get the Security of Multiple Browsers with Just One," in *CCS*, 2011.

[284] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, 1948.

[285] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Institute of Radio Engineers*, 1952.

[286] I. Pavlov, "LZMA specification." http://dl.7-zip.org/lzma-specification.zip, 2013. Accessed on 2013-10-30.

[287] K. G. Morse, Jr., "Compression Tools Compared," *Linux J.*, 2005.

[288] D. Witte, "(doublekey) Key cookies on setting domain * toplevel load domain." https://bugzilla.mozilla.org/show_bug.cgi?id=565965, 2010. Accessed on 2014-07-10.

[289] M. Perry, "Apply third party cookie patch." https://trac.torproject.org/projects/tor/ticket/3246, 2011. Accessed on 2014-07-10.

[290] C. Reis and S. D. Gribble, "Isolating web programs in modern browser architectures," *EuroSys '09*, 2009.

[291] H. Wang, C. Grier, and A. Moshchuk, "The Multi-Principal OS Construction of the Gazelle Web Browser," in *Usenix Securitys Symposiom*, 2009.

[292] C. Grier, S. Tang, and S. T. King, "Secure Web Browsing with the OP Web Browser," *IEEE Symposium on Security and Privacy*, 2008.

[293] A. Eferati, "'Like' Button Follows Web Users." http://online.wsj.com/news/articles/SB10001424052748704281504576329441432995616, 2011. Accessed on 2013-10-30.

[294] D. Fifield and S. Egelman, "Fingerprinting web users through font metrics," in *Financial Cryptography and Data Security*, 2015.

[295] D. Herrmann, K. Fuchs, and H. Federrath, "Fingerprinting Techniques for Target-oriented Investigations in Network Forensics," *Sicherheit*, 2014.

[296] A. FaizKhademi, M. Zulkernine, and K. Weldemariam, "FPGuard: Detection and Prevention of Browser Fingerprinting," *IFIP*, 2015.

[297] C. Torres, H. Jonker, and S. Mauw, "FP-Block: usable web privacy by controlling browser fingerprinting," *ESORICS*, 2015.

[298] A. Khademi, "Browser fingerprinting: Analysis, detection, and prevention at runtime," in *M.S. thesis, School of Computing, Queen's University, Kingston*, 2014.

[299] Gacar, "Circumventing TOR font-limits by using multiple frames." https://www.torproject.org/projects/torbrowser.html.en, 2013.

[300] M. Wu, R. Miller, and S. Garfinkel, "Do security toolbars actually prevent phishing attacks?," *CHI*, 2006.

[301] Appodrome, "CanvasFingerprintBlock." https://chrome.google.com/webstore/detail/canvasfingerprintblock/ipmjngkmngdcdpmgmiebdmfbkcecdndc, 2014. Accessed on 2015-02-02.

[302] The Chromium Project, "Chrome Browser." http://www.chromium.org/Home, 2015.

[303] Tor Project, "Tor Browser." https://trac.torproject.org/projects/tor/ticket/5798#comment:13, 2015. Accessed on 2015-01-11.

[304] H. Tillmann, "Browser Fingerprinting Dataset." http://www.henning-tillmann.de/2013/10/browser-fingerprinting-93-der-nutzer-hinterlassen-eindeutige-spuren/, 2014. Accessed on 2014-09-10.

[305] A. Phillips and M. Davis, "BCP47 - Tags for Identifying Languages," *IETF Trust*, 2009.