Martin Koop*, Erik Tews, and Stefan Katzenbeisser

In-Depth Evaluation of Redirect Tracking and Link Usage

Abstract: In today's web, information gathering on users' online behavior takes a major role. Advertisers use different tracking techniques that invade users' privacy by collecting data on their browsing activities and interests. To preventing this threat, 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. 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. In this work we present the first large scale study on the threat of redirect link tracking. 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 on its tracking ability and demonstrate that top trackers are able to identify the user on the most visited websites. 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.

Keywords: Tracking, Redirect, Privacy, Browser

DOI 10.2478/popets-2020-0079
Received 2020-02-29; revised 2020-06-15; accepted 2020-06-16.

***Corresponding Author: Martin Koop:** Universität Passau, E-mail: mail@martin-koop.de (earlier known Stopczynski)
Erik Tews: University of Twente, E-mail: e.tews@utwente.nl
Stefan Katzenbeisser: Universität Passau, E-mail: stefan.katzenbeisser@uni-passau.de

1 Introduction

It is common practice to use different tracking techniques on websites. This covers the web advertisement infrastructure like banners, so-called web beacons[1] or social media buttons to gather data on the users' online behavior as well as privacy sensible information [52, 69, 73]. Among others, those include information on the user's real name, address, gender, shopping-behavior or location [4, 19]. Connecting this data with information gathered from search queries, mobile devices [17] or content published in online social networks [5, 79] allows revealing further privacy sensitive information [62]. This includes personal interests, problems or desires of users, political or religious views, as well as the financial status. Those valuable information are derived from the website's meta data, such as the content the user is viewing or by the links clicked.

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 first-party tracking elements at each redirected page on the users' machine without notice, circumventing third-party blocking tools. Also, 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 user's interests.

Compared with third-party cookies, first-party cookies are almost always enabled and required by websites to handle sessions. Examples are shopping carts, allowing the user to set preferences (language/currency), or in general, allow the user to login to a service. Blocking all cookies entirely will break the functionality of many websites.

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

In this process, tracking companies store a unique identifier on the user's machine [71] or try to determine an almost unique browser fingerprint of the user's device [46, 59, 62, 63, 71, 74, 83]. Unique identifiers can be archived by cookies, HTML5 localStorage, caching, and common third-party plug-ins like Flash or Java [69]. In consequence, tracking services are able to identify a user across various websites [10, 43] and establish a detailed user profile on web behavior, interests as well as personal information, which creates privacy threats to end-users [20, 63]. These user profiles are then mostly used for targeted advertisement [65], and made available to other companies. Although this data is often anonymized, the information can be reconstructed [20] or easily de-anonymized [6, 62, 66, 79]. Consequently, detailed user profiles may influence credit scores [48] or risk assessments of health insurers to a disadvantage of the user [7]. Furthermore, the loss of anonymity can lead to price discrimination [54, 84], inference of private data [25], or identify theft/fraud. Since third-party tracking is mostly done in the background and without the user's notice, the user has no idea what kind of data is gathered and there is usually no option to opt-out. Studies show that trackers (ad-companies) can track (re-identify) users on about 25-90% of websites [27, 42, 47].

In the last years many browser add-ons such as Privacy Badger[2], Adblock[3], Ghostery[4], or the Tor browser[5] were introduced to mitigate web tracking [53]. Also browser vendors added build-in tracking prevention mechanisms [78]. The capabilities of those anti-tracking methods differ, but the main functionality is in many cases similar: block or isolate third-party elements in order to stop displaying advertisements and prevent them from storing tracking cookies on the user's machine [44, 77].

However, those anti-tracking tools are not sufficient to block all cookie-based tracking techniques, are detectable by the trackers [26, 29, 34, 43, 55, 60, 72, 77] as well as advanced tracking techniques such as browser fingerprinting [3, 9, 14, 53] or as we show in this paper tracking redirect links. Similarly, with the loss of revenue through ad-blockers not displaying advertising [49], many websites implement anti ad-blocker scripts, forcing the user to disable their ad-blocker [38, 58, 64] or use other techniques such as redirect link tracking.

In this work we present the first in-depth study of redirect link tracking focusing on user privacy, giving a comprehensive insight into the behavior, patterns, and used techniques. Our dataset is based on full HTTP traces browsing the top 50,000 websites from the *Alexa.com* ranking[6] over a period of a month. We modified a version of OpenWPM [27], which uses a real Firefox browser that is instrumented using Selenium and included a link selection algorithm, which provides a very realistic browsing experience.

1.1 Redirect Background

Redirects can be used in various scenarios and are mostly utilized to enhance the usability of the web. For example if a website is too slow, not valid anymore, or when the content is changing frequently. In those cases the website can redirect the user to another, valid, new or updated web page. This is commonly used when a URL has moved temporarily or permanently to a new URL. Instead of responding with the requested resource, the HTTP protocol allows the server to respond with a redirect to a new URL by using a corresponding HTTP status code 302/303/307 (moved temporarily) or 301/308 (moved permanent) – see details in RFC7231[7]. The client then tries to load the supplied URL. This is commonly used when a resource has moved from one location to another one or to redirect a client to a server that is geographically closer to the client.

Nowadays, redirects are used for URL shortening or privacy protection by hiding the HTTP referrer when leaving the website. But redirects are also being misused for phishing attacks [1, 22, 81, 85] or to deliver malware through drive-by-download attacks [24, 68, 91]. A website itself may also redirect the browser to a new URL by using for example HTTP meta refresh tag in the HEAD section of the page or by using a JavaScript that instructs the browser to load a new page.

When an HTTP request, that is supposed to load the top-level document of a browser tab or window, is redirected to another domain, or the document itself redirects the browser to another location using JavaScript or a meta refresh tag, it also changes the top-level URL of the browser. If the destination is an-

2 https://www.eff.org/de/node/73969
3 https://getadblock.com/
4 https://www.ghostery.com/
5 https://www.torproject.org/
6 http://s3.amazonaws.com/alexa-static/top-1m.csv.zip
7 https://tools.ietf.org/html/rfc7231

other domain, this changes the first-party domain of the browser.

In this paper, we focus only on redirects that occur while loading the top-level document of a browser window or tab, but not redirects that occur while loading the document for an iframe or while loading a resource for an iframe or the top-level document.

1.2 Redirect Tracking

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

To track a user using redirects, a website that contains links to other parts of the same site or external sites (Figure 1) does not directly link to the target of the link (such as a.com/news) but instead links to another website (*redirect tracker*) (such as b.com/jobs). When the user clicks on the link, he is taken to the redirect tracker first (such as b.com/jobs), which then redirects him to the intended destination (such as a.com/jobs).

Since the redirect tracker is the first-party domain for the browser while the redirect happens, the redirect tracker is able to read or set cookies and since those cookies are first-party domain cookies, they will in general not be blocked.

Optionally, the redirect tracker may also redirect the user through a chain of other redirect trackers (such as c.com/jobs) before he is finally redirected to the intended destination of the link. We call this chain of trackers the *redirect chain* and when the website the link appeared on is the same as the one the user is finally redirected to, we call it a *self redirect*. Where the user is coming from and where he is supposed to be directed too as well as other data can be passed on by the redirect trackers in the HTTP GET parameters and at least the first redirect tracker sees the *Refer* header as well, which is partially also described in [30]. A real example can be found in Section B.

Not every website that performs such a redirect is a *redirect tracker*. Some websites may not do anything at all to track the user. We use the more general term *redirector* for websites that redirect to another domain, without saying whether they also track the user or not.

Since as long as the redirect trackers respond fast, the user experience is the same as clicking on a normal direct link and the user doesn't notice this kind of tracking. In addition, the link target can be hidden using JavaScript or other techniques so that the link looks like a regular link to the intended destination to the user.

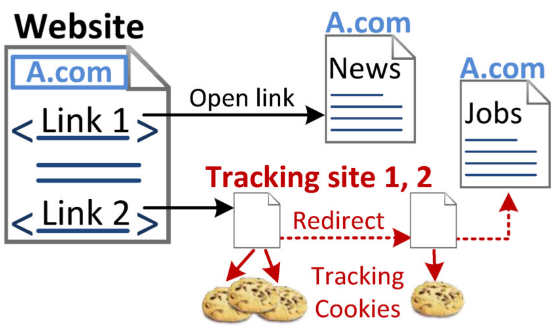

Fig. 1. Redirect link tracking behavior

2 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 [1, 22, 81, 85]. Further, attackers use redirects to deliver malware through drive-by-download attacks [24, 68, 91] and malicious advertising [75, 89, 90]. 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.

Chellapilla et al. [21] also discuss the problem using redirects to provide spam. This includes manipulating search engine crawlers with the goal of improving a website's ranking, and in the context of email, to camouflage spam websites [11].

Li et al. [45] 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 [33] 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 [36] 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 [31, 37, 50, 53, 88] 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 [2, 69, 82]. In contrast to our work, no in-depth studies on redirect behavior on websites were conducted.

Kalavri et al. [41] inspect tracking behavior on user traces collected by a web proxy to explore the resulting tracking graph. They aim 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. [35]. 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 [8]

Schelter and Kunegi [70] 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 contained in cookies. Further, we compare our results to other tracking classifications such as Easylist and OpenWPM [27] to visualize the performance for the reader.

In 2018, Syverson and Traudt reported [80] on the possibility of tracking using HSTS[8], were a cached HTTP connection is redirected to a secure HTTPS connection. Here, the tracker could force the browser to check cached (visited) domains [32]. As of this, browser vendors added additional tracking prevention methods to mitigate HSTS tracking [57, 87]. 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 [51] 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.

The Brave browser team conducted a comparable study [15] focusing on redirects. Crawling the Alexa 10k websites and clicking embedded links, the team found 18,531 websites performing all kinds of redirects. By using a simple heuristic, in which at least two different domains were involved in the redirect, the team identified 365 domains performing first-party redirection-based tracking.

Summing up previous work, even though the idea of tracking the user within redirects is known since 2006 [39] and was discussed in a simple demo in 2011 [18] as well as in [30], 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 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.

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

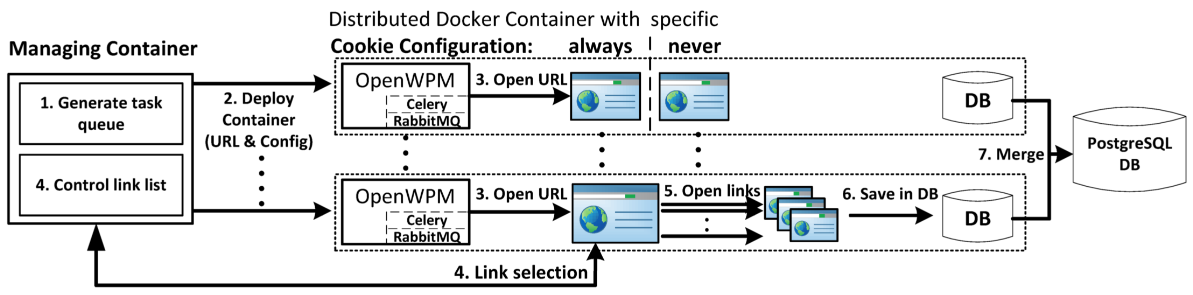

Fig. 2. Experimental set-up

3 Experimental Setup

In order to identify tracking activities, we simulated user browsing behavior within our custom crawler based on OpenWPM [27]. 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 3.4), 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.

3.1 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 an SQLite database.

3.2 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[9], we record if the HTTP channel is being replaced, and store more details about first/third-party contexts in JavaScript. In addition, we trace if a third-party window is being opened 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[10], RabbitMQ[11], and Docker[12]. The crawler architecture is depicted in Figure 2. 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.

9 We added the fields: document_url, top_document_url, referrer_url, and original_url
10 Software for asynchronous task queues: http://www.celeryproject.org/
11 Open source message broker software: https://www.rabbitmq.com/
12 OS-level virtualization to deliver software in packages: https://www.docker.com/

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.

3.3 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 popup 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.

3.4 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 [23]:

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.
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 select 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 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 3 displays the number of external links found on websites, indicating that 10% of the websites have no external links and more than half of the 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 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.

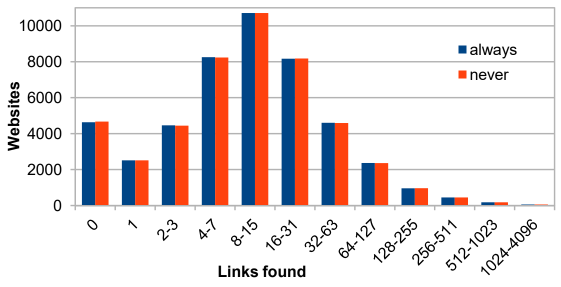

Fig. 3. Amount of links found on each website.

4.1 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.*

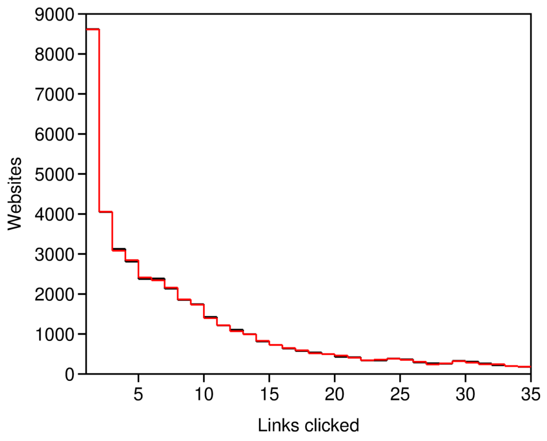

Fig. 4. Amount of links clicked on websites (black: never, red: always).

4.2 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 30 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:

a) Public URL shortening services such as *bit.ly* or *goo.gl*[13].
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[14].
e) Popup-Advertisements[15].
f) Other services that for example redirect a user to a social network (*share*) page.

4.3 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 and their relative ranking. This shows the potential that a tracker can track the users browsing behavior.

Table 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 4.2 are difficult to distinguish 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 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 1 for a comparison). Besides setting third-party cookies, *doubleclick.net* also used redirects in 4% of the cases pointing back to the domain the redirect was triggered. This trick is probably used to set first-party cookies during the redirect and to mitigate ad-blocking plug-ins, that usually block third-party cookies.

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 1. Top 30 regular trackers encountered in our crawl.

4.4 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:

13 Everyone can submit a URL and gets a shortened version of that URL. All shortening services linking to more then 5 different domains are classified as public

14 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.

15 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.

					config always			config never									
Rank	Domain	Class	Redirects	Sites	NV	AP	AC	NV	AP	AC	TC	SR	Alexa	EL	EP	GH	ITP
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 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).

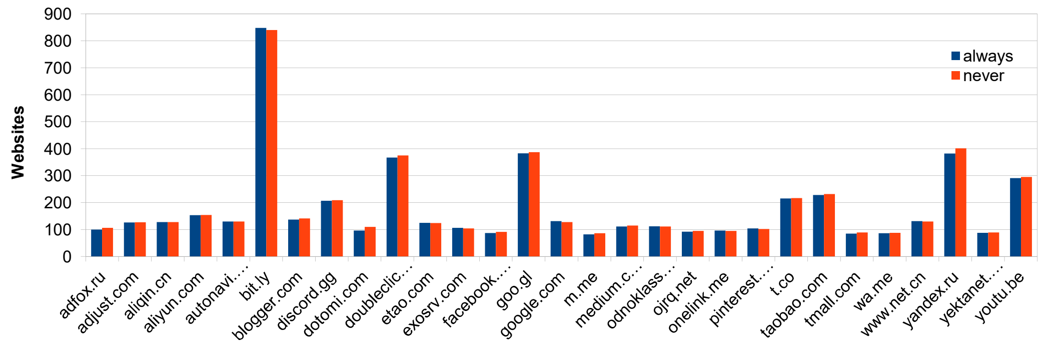

Fig. 5. Top redirect occurrence on different browser settings.

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.

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 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[16], 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) [40]. 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 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 5 displays the results and shows that using more restrictive cookie settings does not affect which redirector we are directed to.

4.5 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

16 https://easylist.to/

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 [27, 28] 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.
- The similarity of the value compared to other values of the same cookie according to the Ratcliff/Obershelp string comparison algorithm[12] 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 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.[17]

4.6 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 3 shows the distribution of the lengths of the chains we encountered according to this definition. To understand this behavior, we build a directed graph of connected redirect domains shown in Figure 6. 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. 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 (37 in the total graph) and *bit.ly* 21 (94 in the total graph) outgoing connection.

We then used the Louvain community detection method [13] 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, including e.g., *buysellads.com*, *carbonads.net*, *smartadserver.com*, *bit.ly*, and others. Table 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

17 Sample cookie exosrv.com: *a%3A1%3A%7Bi%3A0%3Bs%3A33%3A%225e01e859d10810.01004454205715 8735%22%3B* 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.

4.7 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 easyly 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 Appendix in Section B.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 a URL is given in the Appendix in Section B.1.

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.

5 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.

5.1 Rewriting URLs in the Browser

There are a few redirectors for which the destination URL can be easily predicted from the URL parameters (examples are given in the appendix). For those we recommend building 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.

5.2 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 [86], but lacks in many cases as our analysis in section 4.4 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 cookies 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 in section 4.4 shows.

Another effective technique is to automatically isolate the locally stored website state (including cookies, LocalStorage, and browser cache) into separate contain-

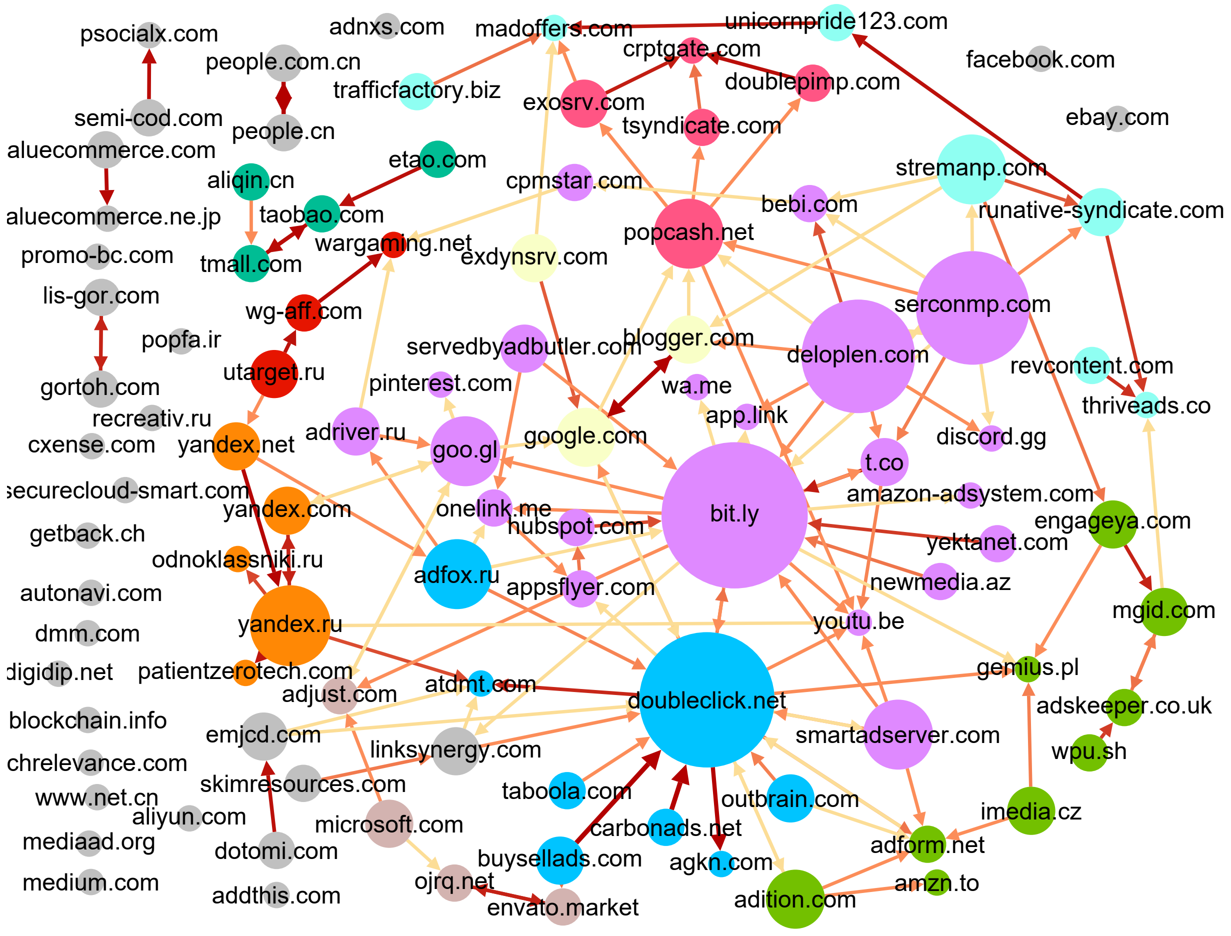

Fig. 6. Dependencies between the redirectors. The width of an arrow from a to b shows how often we have seen a redirect from a to b in our crawls.

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

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. Redirect graph statistic.

ers [77]. 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) [61], it was added to Firefox 52 as part of the Tor Uplift project [56, 67].

Since Firefox 77, the so-called Dynamic First Party Isolation feature was made available to configure the isolation of cookies through the user interface [16]. As discussed in [77], a reason why it was not enabled by default by Mozilla, is that it may break some websites when activated.

5.3 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.

5.4 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.

5.5 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[18] 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[19]. 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 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

18 http://www.greasespot.net
19 http://tampermonkey.net

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.

6 Conclusions and Future Work

In this work 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.

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 [40] as well as Mozilla Firefox [76].

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[20] so that normal users are made more aware of them.

7 Thanks and Acknowledgements

We would like to thank everyone who supported us during this research. The OpenWPM developers, mainly Steven Englehardt and Nihanth Subramanya provided us guidance and feedback when we developed our patches and extensions for OpenWPM so that most of

20 https://privacyscore.org/

them could be integrated into the upstream version of OpenWPM. Both are currently working for the Mozilla foundation and OpenWPM is now an official Mozilla project. We would also like to thank Peter Eckersley who initially pointed us towards the OpenWPM project to conduct this research project.

We would also like to thank the anonymous reviewers of our paper at PETS who provided valuable feedback and improvements that improved the structure and main results a lot. In addition, we would love to thank the Technical University of Darmstadt, Germany, despite none of the authors is currently affiliated with it. The project originally started there and all authors previously worked and/or studied there. Thank you also to Thorben Bürgel, helping in the original project. Furher, we would like to thank the University of Twente, Netherlands, which provided the required computing power that made the large crawls possible. This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

[1] S. Abu-Nimeh and S. Nair. Circumventing security toolbars and phishing filters via rogue wireless access points. *Wireless Communications and Mobile Computing*, 10(8):1128–1139, 2010.

[2] 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, pages 674–689, New York, NY, USA, 2014. ACM.

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

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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, page 37–46, 2016.

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

[11] K. Bhargrava, B. Gsrc, D. Brewer, B. Gsrc, and B. Gsrc. A Study of URL Redirection Indicating Spam. In *CEAS*, 2009.

[12] P. E. Black. Ratcliff/obershelp pattern recognition. *Dictionary of algorithms and data structures*, 17, 2004.

[13] 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*, 2008(10):P10008, oct 2008.

[14] 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*, pages 31–46. Springer, 2012.

[15] Brave. Understanding Redirection-Based Tracking. https://brave.com/redirection-based-tracking/, 2020. Accessed on 2020-04-30.

[16] 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.

[17] J. Brookman, P. Rouge, A. Alva, and C. Yeung. Cross-device tracking: Measurement and disclosures. *Proceedings on Privacy Enhancing Technologies*, 2017(2):133–148, 2017.

[18] 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.

[19] 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.

[20] C. Castelluccia, M.-A. Kaafar, and M.-D. Tran. Betrayed by your ads! In S. Fischer-Hübner and M. Wright, editors, *Privacy Enhancing Technologies*, pages 1–17, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[21] 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, pages 81–88, New York, NY, USA, 2007. ACM.

[22] J. Chen and C. Guo. Online detection and prevention of phishing attacks. In *2006 First International Conference on Communications and Networking in China*, pages 1–7, Oct 2006.

[23] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-Law Distributions in Empirical Data. *SIAM Rev.*, 2009.

[24] 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, pages 281–290, New York, NY, USA, 2010. ACM.

[25] 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.

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

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

[28] 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*, pages 289–299, 2015.

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

[30] 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.

[31] 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)*, pages 151–168, Baltimore, MD, 2018. USENIX Association.

[32] B. Fulgham. Protecting against hsts abuse. https://webkit.org/blog/8146/protecting-against-hsts-abuse/, 2018. Accessed on 2020-02-02.

[33] M. Gandhi, M. Jakobsson, and J. Ratkiewicz. Badvertisements: Stealthy Click-Fraud with Unwitting Accessories. *Journal of Digital Forensic Practice*, 1:131–142, 2006.

[34] 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, pages 259–262, New York, NY, USA, 2017. ACM.

[35] 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)*, volume 1, pages 549–556, Nov 2013.

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

[37] 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*, 2017(1):79–99, 2017.

[38] 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, pages 171–183, New York, NY, USA, 2017. ACM.

[39] C. Jackson, A. Bortz, D. Boneh, and J. C. Mitchell. Protecting browser state from web privacy attacks. In *Proceedings of the 15th international conference on World Wide Web*, pages 737–744, 2006.

[40] John Wilander. Full Third-Party Cookie Blocking and More. https://webkit.org/blog/10218/full-third-party-cookie-blocking-and-more/, 2020.

[41] V. Kalavri, J. Blackburn, M. Varvello, and K. Papagiannaki. Like a Pack of Wolves: Community Structure of Web Trackers. In T. Karagiannis and X. Dimitropoulos, editors, *Passive and Active Measurement*, pages 42–54, Cham, 2016. Springer International Publishing.

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

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

[44] 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.

[45] 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*, pages 1086–1094, 2012.

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

[47] 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*, pages 277–289, 2015.

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

[49] 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, pages 119–125, New York, NY, USA, 2016. ACM.

[50] 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)*, pages 103–116, Baltimore, MD, 2018. USENIX Association.

[51] 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.

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

[53] 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*, pages 319–333, April 2017.

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

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

[56] Mozilla. Tor Uplift Project. https://wiki.mozilla.org/Security/Tor_Uplift, 2017. Accessed on 2020-04-20.

[57] Mozilla. Security/Anti tracking policy. https://wiki.mozilla.org/Security/Anti_tracking_policy, 2019. Accessed on 2020-02-02.

[58] M. H. Mughees, Z. Qian, and Z. Shafiq. Detecting anti ad-blockers in the wild. *Proceedings on Privacy Enhancing Technologies*, 2017(3):130–146, 2017.

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

[60] 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.

[61] 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.

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

[63] 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.

[64] 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, 2016. USENIX Association.

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

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

[67] T. Project. Tor at the Heart: Firefox. https://blog.torproject.org/tor-heart-firefox, 2016. Accessed on 2020-04-20.

[68] 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, pages 1–15, Berkeley, CA, USA, 2008. USENIX Association.

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

[70] S. Schelter and J. Kunegis. On the ubiquity of web tracking: Insights from a billion-page web crawl. *J. Web Science*, 4:53–66, 2018.

[71] 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.

[72] B. Shiller, J. Waldfogel, and J. Ryan. The effect of ad blocking on website traffic and quality. *The RAND Journal of Economics*, 49(1):43–63, 2018.

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

[74] 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.

[75] A. Sood and R. Enbody. Malvertising – exploiting web advertising. *Computer Fraud and Security*, 2011(4):11 – 16, 2011.

[76] Steven Englehardt. Firefox 72 blocks third-party fingerprinting resources. https://blog.mozilla.org/security/2020/01/07/firefox-72-fingerprinting/, 2020.

[77] M. Stopczynski and M. Zugelder. Reducing user tracking through automatic web site state isolations. In *International Conference on Information Security*, pages 309–327. Springer, 2014.

[78] R. Stringham. A Review of Browser Privacy Initiatives and Proposals. https://medium.com/adobetech/a-review-of-browser-privacy-initiatives-and-proposals-4ae86edc23c, 2019. Accessed on 2020-01-02.

[79] 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, pages 1261–1269, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee.

[80] 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, Aug. 2018. USENIX Association.

[81] 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*, pages 180–189, 2010.

[82] 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.

[83] A. Vastel, P. Laperdrix, W. Rudametkin, and R. Rouvoy. Fp-stalker: Tracking browser fingerprint evolutions. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 728–741, May 2018.

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

[85] D. Y. Weider, S. Nargundkar, and N. Tiruthani. A phishing vulnerability analysis of web based systems. In *2008 IEEE Symposium on Computers and Communications*, pages 326–331. IEEE, 2008.

[86] J. Wilander. Intelligent tracking prevention 2.0. https://webkit.org/blog/8311/intelligent-tracking-prevention-2-0/, June 2018.

[87] J. Wilander. Preventing tracking prevention tracking. https://webkit.org/blog/9661/preventing-tracking-prevention-tracking/, 2019. Accessed on 2020-02-02.

[88] 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)*, pages 72–77, Oct 2016.

[89] 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, pages 1286–1295, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee.

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

[91] 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, pages 187–196, New York, NY, USA, 2011. ACM.

A Client Side Redirect Examples

```
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"); ?>
```

B Real redirect example

As an example for real redirect link behavior we visit the website *spiegel.de*. By clicking the 'Job-search' link, the browser will open the following address: *http://pubads.g.doubleclick.net/gampad/clk?id=4529135480&iu=/6032/Clickcommand/clickcommand*, followed by an automatic redirect to:
*http://adclick.g.doubleclick.net/pcs/click?xai=
AKAOjssTn33j5n8gWGUikFv807T6YRmYClOdpi07-
S13C3eLmyYCYiUGZWGdmD8vxNr5q99yrH2ubSzKI-
VEtEUtQAguR6-xYf4O8e560Sk8ieavzxmpzBrPowu-
CYnp3XJOkvdEg3SRrGy37ETfthorWBgYD3l7E0i7j-
HJJZ6EJZprMrAd7quAG8GAzPQonBFxLutCzK1ajES-
zmJLkDr6OQuxIzG5Z0VDkbPvb3LsPG7Fjll97Vyo2-
J0H830aQbZZTG44j&sig=Cg0ArKJSzGNK9pExBeA-
&adurl=http://ad8.adfarm1.adition.com/redi
%3Fsid%3D3076401%26kid%3D1213066%26bid
%3D3962693* and an additional redirect to *http://ad8.adfarm1.adition.com/redi?sid=3076401
&kid=1213066&bid=3962693*. The third redirect will then open the final (legitimate) website: *http://stellensuche.karriere.spiegel.de/*

B.1 Different type of redirect URLs

For example the URL https://share.yandex.net/go.xml
?service=gplus&url=http%3A%2F%2Fkino-history.net
%2F&title=Kino%20History%20-%20%D1%81%D0%B
0%D0%B9%D1%82%20%D0%B8%D1%81%D1%82%D
0%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 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.

The URL https://clck.yandex.ru/redir/dRu0iBr5
YqxShfdSGFXg7CqFK-IM7KfY?data=eE03aEhZNG
duWC1aR1BqRE9uSW1PYUdtOVMzbzV2eUdDYmh
FOHZCbjIwZkpJZTk1TU1yRzNMS0VIWWdJOW5ke
GtrU3RpaXJlcUdVMGJ5QWxiT05sYzV3QjhhUDFE
SzVBSlRMQTFrcEZPWFNFWkFtNm90WElIOVAxV
nBTMHJuLWEtNXNqRkh5eFdlQ2RGNWJScERES0
RSSUN2bnRDS0ZtY3FRNE00MS10TjE4OUdsVVl6a
EJaMko4VngyaFJmV3cwQjNaeVA5c2VHUnQ2bFVC
dDlzYl9xVHhYZkQ2WU80YjlwUGlKZGVlOFBHdkF
uV2dMejF2cDNEdE5hbmFTaC1zbmdhRmlEMmI4V3
BfWjJubXJZeDJJMUVpZzlZdHZKUzVHcUVXVS1jZ
npfbG56TU81Y3hOV3hwX0FnQTVWQ3RJRW5BW
G5RR3JGby16TlFiUFcyMFdHUHozdVZnN1pfdzdXV
URSSjZ0My0xNlB0dE5Xc2psejdlbFA0MzhPb3huU3d
BVnRIZ1VWbExvX0VpOXVVMXpNNUJfRXVtejM
1ODA4VnRGd1JyMUVEbHBVUmZncHlMRkZ2d3JT
M3hQb1Bfa241YVNZRktMUUt3UEVFZXZ5Z05pcjN
hZUFkV25Bb19tcmV5ZUg5R0ZFRXFEX1ZXTGNM
WU8wVlRmV3REWFlVYXQ3QUtqaDJRajM2bTZW
UFlvcUwxN3dRUzBwNFM2Y2pTaFM2NXkyczNOY
mZzbVpjMEJCN0NtRlpjYmpzMnhnYlpQR293NDBj
TkRGeFh2eDNQeW5NRkJuNDREMWVhM0VOYlF
ZTXFTazFQdW9nUmdMLTltM0lmd1hUWkppN0s2O
HB1LTB2TVMyVjBDaklfR2xVYWo5TDBnbWV4Un
JuMmNYTVFkY3B3V1dIZlh2Z01abFZ0c1AxaGFvaX
ZaOTlvVDgtX1NiZXlTbldEVWdGNWlOZXlBanFkO
DhTS0RGWFh4SXl6LWlUV2JwRUppb25vRXR3&b6
4e=2&sign=489b163e63fdd8d4ad5bc56de4b5be68&key
no=3 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.