Block Me If You Can: A Large-Scale Study of Tracker-Blocking Tools

Georg Merzdovnik*, Markus Huber†, Damjan Buhov*, Nick Nikiforakis‡,
Sebastian Neuner*, Martin Schmiedecker*, Edgar Weippl*
*SBA Research, {gmerzdovnik, dbuhov, sneuner, mschiedecker, eweippl}@sba-research.org
†St. Pölten UAS, markus.huber@fhstp.ac.at
‡Stony Brook University, nick@cs.stonybrook.edu

***Abstract*—In this paper, we quantify the effectiveness of third-party tracker blockers on a large scale. First, we analyze the architecture of various state-of-the-art blocking solutions and discuss the advantages and disadvantages of each method. Second, we perform a two-part measurement study on the effectiveness of popular tracker-blocking tools. Our analysis quantifies the protection offered against trackers present on more than 100,000 popular websites and 10,000 popular Android applications. We provide novel insights into the ongoing arms race between trackers and developers of blocking tools as well as which tools achieve the best results under what circumstances. Among others, we discover that rule-based browser extensions outperform learning-based ones, trackers with smaller footprints are more successful at avoiding being blocked, and CDNs pose a major threat towards the future of tracker-blocking tools.**

Overall, the contributions of this paper advance the field of web privacy by providing not only the largest study to date on the effectiveness of tracker-blocking tools, but also by highlighting the most pressing challenges and privacy issues of third-party tracking.

1. Introduction

In the modern internet, it has become a common practice for websites and mobile applications to rely on services provided by third parties. These services include advertisements, analytics, social integration widgets, and CDN-residing versions of popular JavaScript libraries. While the benefits of this third-party content integration are clear for the developers of first-party sites, the widespread adoption of these services is also inevitably linked with increased user tracking.

Every time a user's browser is instructed to fetch a third-party resource, that third-party server is given the ability to deliver tracking scripts and associate the first-party website with the bearer of third-party cookies and browser fingerprints. This tracking of online behavior allows for the construction of increasingly detailed user profiles, including sensitive information such as a user's political views and medical history. In addition to the exposure of users' online behavior to third parties, this third-party communication, which is typically unencrypted, can be further exploited by rogue ISPs and state-level attackers. For instance, it became publicly known that the National Security Agency (NSA) is piggybacking on third-party tracking cookies to de-anonymize Tor users and to identify targets for further exploitation [1], [2].

Third-party tracking thus has serious implications for the overall privacy and security of Internet users. Previous research focused on measuring the prevalence of tracking on common websites [3], [4], [5] and showed how privacy-conscious users and online trackers are at an arms race, with the former deleting their cookies and utilizing client-side, privacy-enhancing technologies, whereas the latter are migrating from traditional stateful tracking to more opaque, stateless tracking technologies based on browser fingerprinting [6], [7], [8], [9].

The absence of explicit policies regarding what a website is and is not allowed to do — coupled with the difficulty of setting and preserving opt-out cookies [10], [11], and the fact that the Do-Not-Track HTTP header is typically ignored by websites [3], [12], [13] — has motivated most savvy users to rely on client-side tools to preserve their online privacy. These client-side tools typically come in the form of browser extensions which differentiate between tracking and non-tracking HTTP requests, blocking the former and allowing the latter. At the time of writing, the two most common blocking tools are *AdBlock Plus* and *Ghostery*. AdBlock Plus focuses on blocking online advertisements, while Ghostery provides feedback on trackers included in websites. Note that even though advertisers do not necessarily need to track user interests in order to show ads, the majority of modern advertisers utilize Online Behavior Advertising which relies on building detailed profiles of a user's interests and is thus one more form of tracking. It is also worthwhile to note that some browser vendors such as Mozilla and Apple have recently acknowledged the importance of tracker-blocking tools and provide native support for rule-based blocking in their browsers [14], [15].

Despite the prevalence of these tracker-blocking tools, there is currently a lack of understanding of their effectiveness and applicability in the wild, and the extent to which they can protect users against motivated trackers. Previous research on the effectiveness of tracker-blocking tools is limited, both in scope as well as their considered threat models [13], [14], [16], [17]. In order to help close that gap,

we present the first large-scale study on the effectiveness of tracker-blocking tools taking into account both stateful and stateless tracking as well as the tracking of a growing number of mobile device users. In particular, this paper makes the following contributions:

- We analyze over 100,000 popular websites and provide an up-to-date view on the reach of online tracking. We find that a small number of companies can effectively track users across the majority of popular websites. We also find that over 60% of tracking information is exchanged over unencrypted HTTP connections.
- We measure the effectiveness of browser extensions in blocking stateful and stateless trackers. We find that the effectiveness among different browser extensions varies, with a small number of extensions effectively blocking the majority of stateful trackers. None of the analyzed extensions was, however, able to block all fingerprinting services.
- We highlight an important research challenge: the lack of effective protection methods on mobile devices. Our analysis discusses the feasibility of blocking trackers on mobile devices based on 10,000 Android applications.

2. Third-Party Tracking

Online tracking typically involves three parties: the host of the online service (the first party), the user (the second party), and the online tracking service (the third party). In our threat model, we account for third-party tracking by both websites as well as mobile applications. This section discusses our threat model in detail.

2.1. Web Tracking

In the following, we describe the two most common web tracking methods, followed by our tracking threat model.

Stateful web tracking. The most commonly used technology to track users online are `persistent cookies` which uniquely identify users across multiple websites. Persistent cookies stay in the user's browser until they either are explicitly deleted by the user or expire. In the case of tracking cookies, the expiration date is set to several years. Next to HTTP cookies, unique identifiers might be also stored in a number of different locations including `Local Shared Objects`, `HTML5 storage`, and `HTTP ETags` [17], [18]. The multitude of possible storage locations for unique identifiers enables persistent user tracking even if HTTP cookies are deleted. As long as the identifiers in one such location escape deletion, they can be used to respawn HTTP cookies [7].

Stateless web tracking (fingerprinting). Stateless tracking methods rely on device-specific information and user-specific configurations in order to uniquely re-identify users. Eckersley [19] conducted the first large-scale study to analyze the uniqueness of web browser configurations, converting them to so called "device fingerprints". Stateless web tracking does not rely on unique identifiers stored on user devices, but on the properties of user devices including: browser version, installed fonts, browser plugins, and screen resolution. Eckersley found that 94.2% of browsers with both Flash and Java installed could be uniquely identified. Follow-up studies by Nikiforakis et al. [6] and Acar et al. [7], [8] showed that stateless web tracking is already used in the wild. Englehardt and Narayanan [9] recently showed that fingerprinters are expanding their arsenal of techniques, e.g., with audio-based fingerprinting.

Next to the utilized tracking method, web tracking also depends on how third-party content is integrated into websites. *Analytics services* such as Google Analytics are included as third-party scripts, and thus set a unique identifier per site and user. As such, these services typically do not have globally unique identifiers per user. *Advertisement services* are typically included within an `iframe`, and advertisement providers can therefore set a global (i.e., site-independent) tracking identifier per user. *Social widgets* act as a first- and third-party and can therefore track users uniquely across multiple websites. An example of a social widget is Facebook's "Like" button, where a unique per-user cookie is set by Facebook and transmitted back to Facebook from all websites that display that button. Trackers that have both first-party and third-party roles have other user information in addition to unique identifiers such as the users' full names and e-mail addresses. First parties might also unintentionally leak personal information to third parties via various coding mistakes [13], [20].

2.2. Mobile Tracking

The tracking threats involved in browsing websites apply equally well to desktop browsers as well as browsing done via mobile devices. Mobile devices, however, provide an additional source of potential information leakage: third-party services bundled with mobile applications [21], [22], [23]. In-app third-party services transmit unique device identifiers (`UDID`) to track the behavior of users. In older versions of Android and iOS, third parties were allowed to access these immutable UDIDs. Since 2013 both mobile operating systems replaced immutable UDIDs with advertisement IDs which can be reset by users. Mobile third-party trackers can still, however, collect additional unique device identifiers such as the device's `IMEI` or `MAC` address of the WiFi interface in order to reclaim reset-resistant tracking.

2.3. Threat Model

We exclude first-party tracking from our model under the assumption that users visit those sites intentionally and the sites may legitimately need to track users to provide, for instance, the notion of a session. Our threat model accounts for the following threats posed by third-party trackers:

- **Stateful and stateless tracking by third parties**
  Both stateful and stateless tracking methods ultimately rely on transferring either a unique user identifier or a device fingerprint to the tracking third party.
- **Passive collection of transmitted identifiers**
  The passive collection of transmitted identifiers is enabled by additional third-party trackers who rely on unencrypted communication protocols.

In this paper, we investigate the interactions of the trackers described in our aforementioned threat models with tracker-blocking software, because it is currently the only protection method which prevents communication with third-party tracking services. Furthermore, we use a more stringent classification of tracking services than that of prior work, e.g., compared with the work of Roesner et al. [3]. Specifically, we do not treat analytics services separately, because we argue that these services can hypothetically match per-site identifiers to a unique user profile based on various system properties (fingerprints) they collect. However, our threat model does not account for Internet Service Providers (e.g., [24], [25]) or other entities that are able to actively manipulate en route web traffic.

3. Blocking Trackers

In this section we describe the various tracker-blocking methods that users have at their disposal, including the ones that can access and block all network traffic as well as the ones specifically situated in users' browsers.

3.1. Network-based blocking

Network-based blocking methods were in use long before web browsers supported the notion of plugins and extensions. In the following, we discuss the most common network-based filtering methods and their drawbacks.

DNS blocking. DNS blocking uses address-based blacklists in order to block access to certain domains. In the context of blocking trackers (including ads) DNS blacklists are commonly distributed in the form of a `hosts` file. These hosts files are intended as replacements or extensions to the stock hosts files of operating systems. A number of projects maintain hosts files with popular advertising and tracking domains which redirect requests to these domains to `localhost`. Examples of DNS blacklists focused on blocking advertisements and trackers include the longstanding *MVPS hosts* [26] and *Peter Lowe's list* [27]. DNS blacklists exist since the late 1990ies and are now experiencing a renaissance for blocking in-app advertisements on rooted mobile devices [28]. This tracker-blocking method works independently of the used application, but is also the most coarse-grained form of blocking. DNS filtering can be used to block entire (sub)domains, but not individual `URIs`. That is, one cannot block *newyorktimes.com/tracker.js* while maintaining access to *newyorktimes.com*.

TABLE 1. COMMON BROWSER EXTENSIONS TO BLOCK ONLINE TRACKERS, INSTALLATIONS, AND UNDERLYING FILTER RULES (AUG. 2016).

Browser Extension	Filter-Rules	Firefox Users	Chrome Users
AdBlock Plus (ABP)	ABP	18,689,656	10,000,000+
AdBlock	ABP	NA	10,000,000+
Ghostery	custom (proprietary)	1,337,831	2.348.209
uBlock (Origin)	ABP	1,243,409	3,852,990
AdBlock Edge	ABP	408,410	NA
Disconnect	custom (GPL)	265,773	797,097
Blur	custom (proprietary)	176,027	329,446
Privacy Badger	algorithmic	80,291	324,062

Interception Proxies. Interception proxies forward and modify web traffic. A popular privacy-enhancing interception proxy is *Privoxy* [29]. Privoxy has URI-based filtering capabilities and can modify the content and headers of web requests. As such, interception proxies can be used for more fine-grained blocking of third-party tracking by removing individual cookies, blocking certain URIs, and removing tracking code from web pages. Interception proxies cannot, however, intercept or modify encrypted `TLS` (HTTPS) traffic. In theory, interception proxies could use their own custom TLS Certification Authority to modify HTTPS traffic. In practice, an active `man-in-the-middle` attack that blocks trackers puts users at a serious security risk [30]. Furthermore, even with a custom Certification Authority, interception proxies cannot handle websites that utilize certificate pinning [31], [32].

All network-based blocking methods have the advantage of working independently of the underlying application or browser. These methods have, however, two important shortcomings: First, as mentioned earlier, they cannot perform fine-grained blocking on encrypted web traffic (proxy), but only block entire domains (DNS based). Second, third-party content cannot always be reliably detected at a network level. Specifically, at every third-party request, the network-level blocking tool must be able to reliably differentiate intentional third-party requests (user clicked on a third-party link and is expecting to navigate to a different website) from unintentional, tracking-related, third-party requests.

3.2. Browser Extensions

Browser extensions can reliably detect third party content and modify any content loaded by web browsers including encrypted web traffic. Table 1 shows the most popular tracker-blocking browser extensions available for users of Mozilla Firefox and Google Chrome. In the following, we describe the different browser extensions in more detail.

Ad Blockers. The apparent need for blocking ads has led to some of today's most popular browser extensions. This trend also becomes apparent when comparing the install counts of different tracker-blocking tools across extension markets. *AdBlock Plus (ABP)* is the most popular of these extensions, at the time of writing ABP also has by far the most users of all Firefox extensions. ABP limits user tracking by blocking

ads from being loaded. A number of other extensions build upon the filter rules by ABP (shown in Tab. 1). ABP filters are written using a custom pattern syntax and are then internally translated to regular expressions. There are two basic types of ABP filter rules: general blocking filters and `CSS` filters. CSS filters are used to hide previously blocked ad elements on websites. In addition to filter rules, exceptions for these filters can be defined. The use of regular expressions in the filter rules is discouraged, because of the potential performance impact. By default, ABP subscribes to *EasyList* filter rules which include general adblocking rules. A number of additional subscriptions exists to improve regional blocking of ads as well to block additional third-party trackers (*EasyPrivacy*).

Tracker Blockers. Tracker-blocking extensions focus on blocking trackers. The most popular extension in this category is *Ghostery*. It is important to note that Ghostery does not block trackers by default, but merely provides feedback on which third-party trackers are included in each visited website. Similar extensions are *Disconnect*, Abine's *Blur*, and EFF's *Privacy Badger*. Tracker blocking rules include: third-party domains, specific URIs, and "surrogates." Surrogates offer click-to-play functionality for social widgets similar to the ones proposed by Roesner et al. [3].

3.3. Different Types of Rulesets

The effectiveness of all tracker-blocking methods discussed so far depends on their underlying blocking ruleset. Rulesets can be divided into three categories: *community-driven*, *centralized*, and *algorithmic*. The most popular community-driven rulesets for blocking ads and trackers origin from the development of the AdBlock Plus browser extension. At the time of writing, the main AdBlock Plus ruleset (`EasyList`) consists of over 17,000 URI patterns and more than 25,000 CSS tags to be blocked. `EasyPrivacy` is a ruleset for AdBlock Plus with more than 9,000 community-maintained rules targeted at blocking trackers. The subscriptions offered by the AdBlock Plus community are used in a number of other browser extensions, including AdBlock, AdBlock Edge, and uBlock. Every change to the AdBlock Plus rulesets is tracked via a public mercurial repository [33]. Eyeo, the company behind the AdBlock Plus browser extension, started an "acceptable ads program" at the end of 2011 [34]. By 2015 Eyeo's acceptable ads program allowed for the whitelisting of over 300 businesses [35]. The acceptable ads program is enabled by default for the AdBlock Plus browser extension.

Ghostery, Disconnect and Blur rely on a centralized approach to create blocking rules. This means that the companies behind these three tracker-blocking tools maintain and curate blocking rules. These centralized, top-down filter rules are, in general, more compact than community-driven approaches. For example, Disconnect consists of a list of 2,200 third-party domains, whereas there are over 9,000 rules in the community-driven EasyPrivacy ruleset.

The third category are algorithmic approaches for blocking rules. These blocking tools do not rely on regularly updated blacklists, but instead use heuristics to automatically detect third-party trackers. The most popular example for the use of algorithmic rulesets is EFF's Privacy Badger which labels third parties as trackers by observing the requests between first-party and third-party websites and searching for the same high-entropy strings exchanged between multiple first-party websites and individual third-party ones.

4. Methodology

In this section, we describe the methodology of our large-scale tracker analysis. The section has two main parts: our web tracking evaluation and our mobile tracking evaluation.

4.1. Web Tracking

We evaluate the effectiveness of the most popular rule-based advertisement and tracker blocking browser extensions. Specifically, we use the following browser extensions:

- AdBlock Plus 2.7.3 (default settings)
- Disconnect 3.15.3 (default settings)
- Ghostery 6.2.0 **(blocking activated)**
- EFF Privacy Badger 0.2.6
  (trained with Alexa Top 1,000)
- uBlock Origin 1.7.0 (default settings)

Overall, we use, whenever possible, browser extensions with their default settings to simulate the experience of users who install an extension, but do not further configure them. We include AdBlock Plus, because it is by far the most popular browser extension to block advertisement which is one form of tracking due to the trend of Online Behavior Advertising (OBA). AdBlock Plus by default relies on the EasyList ruleset, but whitelists some "acceptable" advertisement networks. Ghostery is the most popular browser extension focused on online tracking, but displays by default only detected trackers and does not block them. For our measurements, we thus activated blocking for all of Ghostery's third-party categories. Disconnect is an alternative for Ghostery, and Disconnect's ruleset is also used for Firefox's tracking protection [14]. uBlock Origin markets itself as a lightweight alternative for AdBlock Plus and a "wide-spectrum blocker". We use uBlock with its default settings — which include EasyList and EasyPrivacy rulesets — and other community-driven rulesets for blocking: ads, trackers, and malware. Finally, we trained EFF's Privacy Badger with the Alexa Top 1,000 websites to evaluate the effectiveness of this novel algorithmic blocking approach.

Analysis Framework. In order to analyze the browser extensions, we developed a distributed modular web crawler framework called CRAWLIUM. We designed the CRAWLIUM measurement framework, because none of the existing frameworks, such as OpenWPM [36], were able to run multiple browser configurations in parallel and support high scalability at the same time.

A high-level system overview of our framework is outlined in Figure 1. Our analysis framework

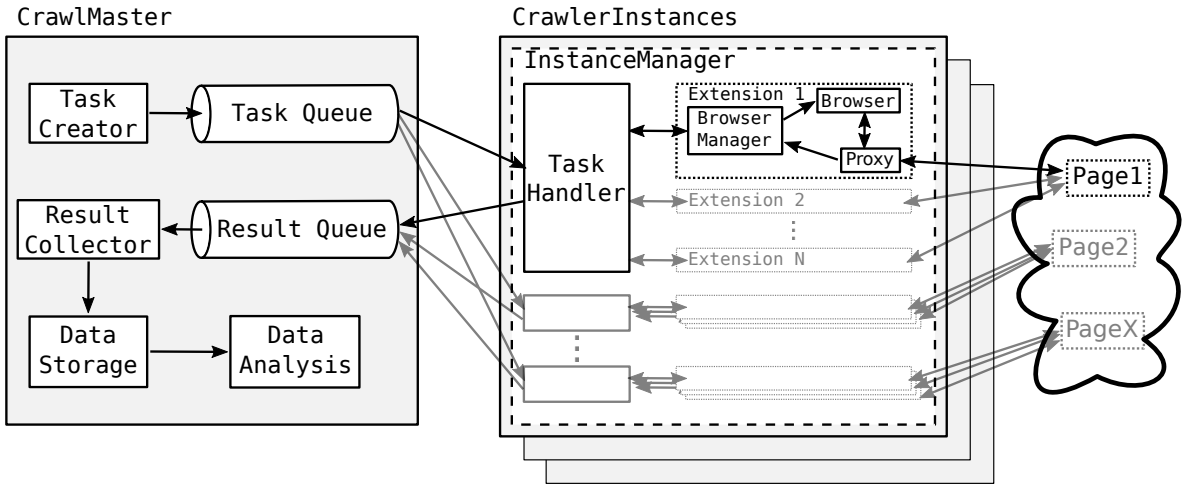

Figure 1. System overview of our modular web measurement framework CRAWLIUM. The CrawlMaster is responsible for task distribution and data handling. The CrawlerInstances each manage several task handlers which process the tasks by running them against the different browser extensions in parallel. The results are sent back to the CrawlMaster for collection and analysis.

consists of two instance types: `CrawlMaster` and `CrawlerInstances`. The `CrawlMaster` instance is responsible for producing tasks and aggregating results, while the `CrawlerInstances` are responsible for executing the actual measurement tasks. The `CrawlMaster` needs to run all the time for uninterrupted operation, whereas `CrawlerInstances` can be terminated at any point in time. Each crawler instance hosts several `TaskHandlers`. They are responsible for running the tasks on the browsers and sending the results back to the manager instance. Each `TaskHandler` is responsible for a set of `BrowserManagers`, one for each browser extension and another one without modifications, as a baseline for the analysis ("*plain*" profile). BrowserManagers handle command execution on browsers and collect results from the proxy. They also take care of restarting browsers and proxies and retry execution of tasks in the case of an error. By running the browser with the different profiles in parallel, we account for potential temporal effects during crawls. These temporal effects are, for example, changes to a website between requests from different profiles. Since our setup is very modular it can be easily adapted for other user profiles. The actual framework is based on Firefox, Selenium and mitmproxy [37] in order to load web pages and collect request and responses.

Im order to thoroughly evaluate the extensions concerning their functionality, we also collect statistics on resource consumption. Therefore, our framework collects information on CPU usage and memory consumption for each browser based on a task granularity. Every time before a new task is sent to the browser, the BrowserManager collects the current memory consumption and resets the count for CPU usage. After the task is handled by the browser we store the percentage of used computation resources and collect a second memory snapshot. Finally, for each finished task we reset the browser by closing all windows and pages that were opened during crawling.

Web Sample. The sample of our evaluation is seeded from the global *Alexa Top Sites*. The detection requires the analysis of the actual (sub)pages from the Alexa Top Sites which contain trackers. On certain websites, such as news sites, tracking code (e.g., social widgets) is embedded in news pages and not on the landing page of domains. We therefore use a twofold crawling strategy to account for trackers on nested web pages. First, we use `PhantomJS` to determine the landing page of the top 200,000 domains in the global Alexa Top Sites dataset. This phase of the measurement does not collect final results, but serves as a first stage to determine the sample set of pages for the second stage of the measurement. We chose PhantomJS to account for `AJAX` requests and thus have access to the content of current websites, while still maintaining a low profile during sample set selection. Second, we enumerate all nested links on a given domain by analyzing the gathered websites. After collecting the nested links, we select two random subpages for each website. If no nested links are found or the domain did not respond, the given domain is excluded. Finally, each valid website sample consists of three webpages: the landing page and the two random subpages.

Domain Aggregation. In order to extract the main domains from URIs, we rely on the *python tld package*, which in turn uses Mozilla's public suffix list [38]. We removed user-provided TLD information from Mozilla's list to accurately group our results based on providers instead of individual services.

Detection of Fingerprinters. In addition to traditional, stateful third-party tracking, our large-scale evaluation accounts for tracking based on fingerprinting. Our analysis is based on the findings provided by Acar et al. in FPDetective [8] and Englehardt et al. based on OpenWPM [9]. Acar et al. provide several regular expressions [39] to detect fingerprinters based on their URIs, while Englehart et al. provide specific URI's to identify fingerprinters. We used

these regular expressions and URI's to detect if a page includes a fingerprinting script based on the collected results. This analysis provides a baseline on the effectiveness of existing browser extensions to prohibit the execution of well-known and recently identified fingerprinting scripts.

4.2. Mobile Tracking

To better understand the difference between desktop and mobile systems we analyzed a sample set of 10,000 popular Android applications concerning the inclusion of third-party trackers. We obtained the sample from Viennot et al. [40]. We expected that the browser extensions ported to mobile platforms yield comparable results to our web tracking experiments. This analysis complements our web tracking experiments in the sense that we include in-app third-party services which cannot be easily blocked without rooting the underlying operating system. We conducted pilot experiments by statically analyzing our collected sample with androguard [41] to extract activities found in the applications' manifest file. These pilot experiments, however, showed a clear drawback of static analysis: Although the extracted activities point to a specific ad provider, we were unable to determine the ad-network tracking mobile users. The main reason behind the limit of static analysis regarding mobile in-app tracking is, that nowadays most mobile ads are delivered through mediation networks such as Google AdMob [42]. This implies that even if an application contains an activity pointing to the AdMob advertisements, it does not tell us whether the advertisement which is going to be served will originate from AdMob itself or from some different ad provider which is part of the mediated networks.

To overcome the limitation of static analysis, we set up a dynamic analysis environment. Our framework is executed in the Genymotion emulator [43] and exercised apps with the help of Monkeyrunner. We furthermored use MITM proxy to intercept all outgoing and incoming traffic for each application in our sample. Finally, we evaluate two common blocking approaches for mobile in-app advertisement: DNS-based blocking and *Adblock Plus for Android*. In order to analyze DNS-based blocking, we use the rulesets of two different applications, *AdAway* [28] and *MoaAB (Mother of all AD-BLOCKING)* [44]. AdAway and MoaAB are Android applications for rooted Android phones which rely on a network-based approach to block mobile in-app advertisement by replacing Android's stock host file. The Adblock Plus for Android application does not require root access, but creates a proxy which filters web traffic based on the same filter rules as the Adblock Plus browser extension. The Adblock Plus Proxy only intercepts HTTP traffic and thus does not block advertisements on HTTPS requests [45]. In order to evaluate the effectiveness of these three popular mobile blocking tools, we matched their underlying rulesets against the collected requests from our dynamic Android application analysis. For AdAway and MoaAB, we use their hosts files, for Adblock Plus we match requests against their default ruleset.

5. Results

In this section we present the results of our evaluation of more than 100,000 popular websites and 10,000 Android applications.

5.1. Collected Data

We seeded our crawling framework with the top 200,000 Alexa websites. The actual crawling was performed on Amazon EC2 via their Oregon datacenter, therefore all requests originated from the same region in order to limit potential location bias. We carried out multiple measurements of the Top 200,000 Alexa websites to ensure our framework produces comparable and valid samples. The data discussed in this section is based on a sample obtained in May 2016. Out of 200,000 websites from the Alexa dataset, we consider 61.93% of them as having been properly crawled. This rather low number has two reasons: First, during the PhantomJS analysis a number of websites did not respond, which reduced the set of tasks available for the second stage of the crawl to 191,492 websites (4.25% failed). Second, we only considered web site samples where none of the browser extensions caused timeouts when loading. The number of successful results for the different extensions is shown in Table 2. The use of Privacy Badger caused the highest number of failed samples, where only 71% of all websites loaded without time-outs with this extension installed. Except from Privacy Badger, all extensions improved our sample collection success rate compared to the plain profile, while less than 10% of all websites produced timeouts. Our filtering process finally resulted in a total set of 123,876 websites which were successfully analyzed with all browser extensions. These websites are uniformly spread in the Alexa top 200K ranks; therefore, we argue that our results are generalizable and characteristic of the entire range. In terms of requests, our crawlers were able to collect over 137 million HTTP(S) requests.

TABLE 2. SUCCESSFULLY CRAWLED WEB PAGES PER EXTENSION. WE CONSIDER A REQUEST PER EXTENSION AS FAILED IF THEY EITHER DID NOT RETURN ANY RESULTS AT ALL AFTER 3 TRIES OR HAD AT LEAST ONE EMBEDDED REQUEST TIME-OUT AFTER 90 SECONDS.

Plugin	# Sites	# Success	Failed %
plain	191,492	164,815	13.93%
adblockplus	191,492	170,636	10.89%
disconnect	191,492	176,659	7.75%
ghostery	191,492	179,068	6.49%
privacybadger	191,492	136,796	28.56%
ublock-origin	191,492	178,233	6.92%

The analysis of Android applications was performed at our local lab. We used our dynamic analysis framework to collect the network requests of the 10,000 most popular Android applications. We excluded 939 applications from our set that caused runtime errors while being analyzed.

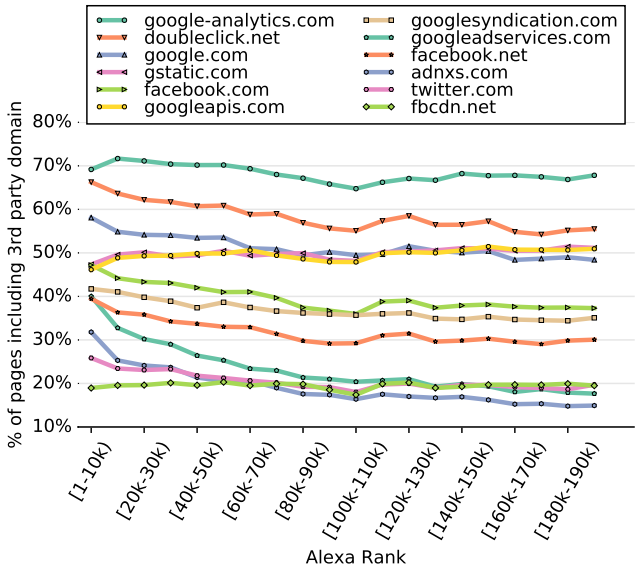

Figure 2. Distribution of most popular third-party domains (TLD+1) in Alexa Top 200,000 websites in 10,000 intervals.

5.2. Identified Third-Party Services

We extracted the set of domains to which the different browser instances issued requests. The actual top level domains were identified based on Mozilla's public suffix list (see subsection 4.1). The information was then aggregated to determine how often a specific domain (TLD+1) occurs in different popularity ranks of first-party websites. Figure 2 outlines the distribution of the most popular third-party domains in our crawled set of popular websites.

We found that the great majority of third-party services belongs to a relatively small number of large Internet players. Table 3 shows an aggregated view on the third-party services we observed, based on the meta-information of Falahrastegar et al. [46]. Google provides by far the most popular third-party web services; overall, *Google services are included by ∼97% of websites in our sample*. The reach of Google can be attributed to three service categories: analytics, advertisements, and CDN services (e.g., googleapis.com). Social widgets by Facebook are included by ∼47% and those of Twitter by ∼24% of all websites in our sample. Amazon is the third-biggest third-party service due to their CDN and cloud computing services.

Mobile Third Parties. Table 3 also shows the reach of Internet companies in the context of Android applications. Google's reach of 74% of their third-party services for Android applications follows intuitively as Google develops Android. In comparison with our web sample, however, it appears that Facebook, Twitter and Amazon have considerable less reach on Android applications in comparison to websites.

Insecure Content Delivery. Our results indicate that the majority of observed third-party services use plaintext protocols to deliver content and to exchange tracking information. Figure 3 outlines the inclusion of third-party content through

TABLE 3. PERCENTAGE OF WEBSITES AND ANDROID APPLICATIONS REACHED BY THE TOP 15 COMPANIES THAT PROVIDE THIRD-PARTY SERVICES. THE RESULTS SHOW THE TOTAL REACH (PLAIN) AS WELL AS THE REACH AFTER THE APPLICATION OF EACH BLOCKING SOLUTION. FOR THE WEB DATASET THESE ARE ADBLOCK PLUS [ABP], DISCONNECT [DC], GHOSTERY [GH], PRIVACYBADGER [PB], UBLOCK ORIGIN [UBO], AND ALL APPLICATIONS COMBINED [C]. FOR ANDROID THESE ARE EASYLIST [E], ADAWAY [A] AND MOAAB [M]

	Desktop							*Mobile*			
	plain	abp	dc	gh	pb	ubo	c	plain	e	a	m
Google	97	93	80	66	93	69	60	74	74	57	54
Facebook	47	44	5	2	4	39	0	6	6	6	6
Amazon	25	21	21	13	20	13	10	8	8	8	7
Twitter	24	21	6	1	19	19	1	1	1	1	1
Yahoo	18	6	4	2	3	2	1	14	14	14	0
AddThis	15	14	8	0	0	0	0	0	0	0	0
ComScor	14	10	1	0	1	0	0	2	2	0	0
AOL	11	0	1	0	1	0	0	0	0	0	0
Adobe	10	5	0	0	0	0	0	0	0	0	0
Quantcast	9	5	1	0	0	0	0	0	0	0	0
Conversant(ValueClick)	8	1	0	0	1	0	0	0	0	0	0
RadiumOne	6	1	0	0	0	0	0	0	0	0	0
Baidu	6	6	6	2	0	1	0	2	2	2	0
AudienceScience	5	0	0	0	0	0	0	0	0	0	0
Sizmek	5	0	0	0	1	0	0	0	0	0	0

`HTTP` and `HTTPS` for our complete set of analyzed websites. Our results show that more than 60% of websites still use HTTP for third-party content delivery. There are also several web pages that access the same third-party domain through HTTP and HTTPS. This behavior is likely due to initial requests performed through HTTP and upgraded to HTTPS by a third-party content provider. We found that, for our Android sample, more than 75% of all requests were performed over HTTP. Furthermore, the mobile blocking tools we analyzed had an overall small impact on requests to third-party services, with the best DNS-based blocking list (MoaAB) reducing requests to third parties by 25%. As such, our results effectively show that browser extensions reduce insecurely loaded third-party content on websites. Ghostery, for example, limits insecurely loaded third-party content to about 20%. This, however, means that attackers could, in the worst case, still target users on every fifth website through passive and active attacks on third-party services.

5.3. Blocking Behavior and Shortcomings

We categorized the 30 most popular third-party tracker domains identified in our experiment to compare the effectiveness of our analyzed browser extensions with respect to blocking specific third-party service categories. Figure 4 shows the effectiveness of browser extensions in blocking the most common categories of third-party services. This figure outlines the blocking behavior of the popular browser extensions we analyzed. Table 6 in the Appendix provides a detailed view on the 30 most-popular third parties identified in our measurements, and the impact of tracker-blocking browser extensions. For example, the first row in Table 6 shows that Google Analytics was the most popular third-party service in our sample, with 53.6% of requests

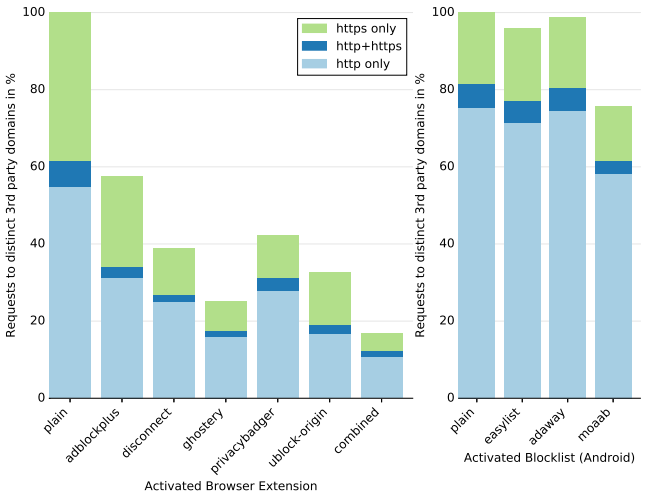

Figure 3. Protocols used for requests to distinct third-party domains.

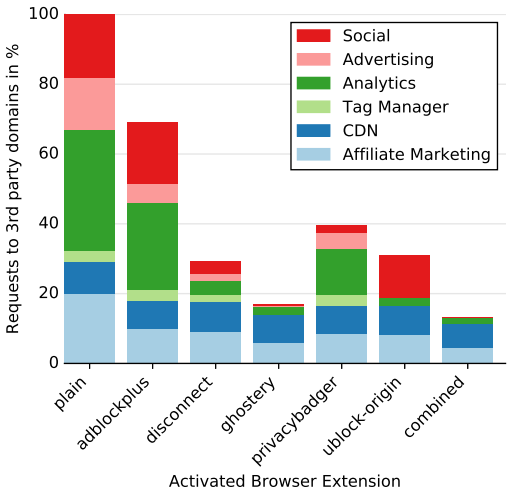

Figure 4. Categories blocked by different extensions, and all extensions combined. The data shows the categorized and aggregated numbers of the 30 most popular third-party services in our sample of 123,876 websites.

to *www.google-analytics.com* performed over HTTP, and 13.2% performed over HTTPS (plain column). The columns following "plain" show the impact of our evaluated browser extensions where, for example, uBlock effectively blocks all HTTP and HTTPS requests to Google Analytics.

AdBlock Plus is by far the most popular ad blocker, and our findings show that **AdBlock Plus blocked the least amount of advertising-related, third-party requests of all tracker-blocking extensions** in our measurements. The blocking behavior of AdBlock Plus can be attributed to its acceptable ads program (discussed in Section 3) which resulted in an overall decrease by 4% of blocked advertisements in our measurements. Table 6 details our findings: the majority of browser extensions, e.g., completely block *googleads.g.doubleclick.net*, while Adblock Plus still allows it to be included in about 1.5% of pages through HTTP and 13.7% of pages through HTTPS.

Furthermore, our measurements highlight an important issue: **a number of browser extensions fail to effectively block social widgets** (e.g., Facebook's "Like" button or Twitter's button) from tracking users. Disconnect fails to block requests originating from Twitter's social widgets in our measurements. uBlock Origin, with the community-driven EasyPrivacy rules, fails to significantly impact tracking by major social networks such as Facebook or Twitter. PrivacyBadger is the only extension to completely block third-party requests to *https://www.facebook.com* (see Table 6) but does not block all requests to Twitter.

Overall, our results suggest that top-down approaches for rulesets (Disconnect, Ghostery) outperform community-based rulesets (Adblock Plus, uBlock Origin) in terms of their overall effectiveness in blocking the most popular third-party trackers. *PrivacyBadger* takes a different approach at blocking third-party trackers and does not come with a preloaded blocking list, with its blocking capabilities depending largely on the pages that have been previously visited. Furthermore, certain major third parties, like *Google Analytics*, are not considered trackers, because they do not share state between pages and are therefore not blocked by *PrivacyBadger* at all [47]. Overall, PrivacyBadger showed promising effectiveness when compared with traditional rule-based blocking extensions, but, as previously discussed, also led to a large number of timeouts and therefore to a potentially large number of malfunctioning webpages.

Mobile Blocking. Table 4 outlines third-party services detected in our Android application sample and the impact of our evaluated blocking tools in detail. The three blocking tools we evaluated offered limited protection against third-party tracking. AdAway [a] blocked the four most popular domains for mobile ad delivery (DoubleClick, Ads by Flurry, googlesyndication, and admob), but did not include rules for *Chartboost* and other common mobile analytics providers. Adblock Plus for Android with their default EasyList [e] ruleset detected common advertisement providers, but this proxy-based solution can not block HTTPS requests. Table 4 highlights this limitation of Adblock Plus for Android. For example, the first row of Table 4 shows that 41.89% of apps that make requests to googleads.g.doubleclick.net are not blocked, because they used HTTPS. The default rulesets of Adblock Plus also lack mobile-specific rules for other popular third parties such as AdMob. The DNS-based "Mother of All AD-BLOCKING" [m] blocklist had the biggest impact on third-party tracking in Android applications. There are **specific third-party services that none of these tools can easily block**. Facebook, for example, uses HTTPS for all requests, meaning that proxy-based blocking does not work, and DNS-blocking of, e.g., *graph.facebook.com* would break the functionality of applications.

5.4. Blind Spots of Different Rule Sets

Apart from the ability of browser extensions to block the third-party domains with the largest footprint in terms

TABLE 4. PER-APP DISTRIBUTION OF THE TOP 15 THIRD-PARTY SERVICES IN THE ANDROID SAMPLE IN % OF TOTAL APPS AND THE IMPACT OF DNS-/PROXY- BASED BLOCKING.

		plain	easylist	adaway	moaab
		http/https	*http/https*	*http/https*	*http/https*
doubleclick.net	googleads.g	17.22/41.89	/41.89	/	/
	stats.g	0.08/0.71	0.08/0.71	/	/
	pubads.g	0.26/0.43	/0.43	/	/
google.com	android.clients	0.08/28.48	0.08/28.48	0.08/28.48	0.08/28.48
	www	2.37/4.97	2.24/4.97	2.37/4.97	2.37/4.97
	accounts	/0.61	/0.61	/0.61	/0.61
googlesyndication.com	pagead2	12.56/15.52	/15.52	/	/
	tpc	1.38/11.60	0.03/11.60	/	/
	video-ad-stats	0.01/	0.01/	/	/
googleapis.com	fonts	7.12/11.09	7.12/11.09	7.12/11.09	7.12/11.09
	www	/4.44	/4.44	/4.44	/4.44
	play	/4.35	/4.35	/4.35	/4.35
gstatic.com	fonts	6.72/10.74	6.72/10.74	6.72/10.74	6.72/10.74
	csi	2.22/3.62	2.22/3.62	2.22/3.62	/
	www	3.07/0.98	2.91/0.98	3.07/0.98	3.07/0.98
admob.com	media	16.52/0.22	16.52/0.22	/	/
	e	0.02/	/	0.02/	/
google-analytics.com	ssl	/11.44	/11.44	/	/
	www	3.92/0.63	3.92/0.63	/	/
googleusercontent.com	lh3	6.38/8.38	6.38/8.38	6.38/8.38	6.38/8.38
	lh5	0.03/0.28	0.03/0.28	0.03/0.28	0.03/0.28
	lh4	0.06/0.15	0.06/0.15	0.06/0.15	0.06/0.15
flurry.com	data	9.30/3.94	9.30/3.94	9.30/3.94	/
	ads	0.26/0.73	/0.73	/	/
	cdn	/0.66	/0.66	/0.66	/
adobe.com	mobiledl	8.77/	8.77/	8.77/	8.77/
	airdownload2	6.91/	6.91/	6.91/	6.91/
	sp.auth	/0.09	/0.09	/0.09	/0.09
chartboost.com	live	/4.34	/4.34	/4.34	/
	a	/3.16	/3.16	/3.16	/
	www	/3.09	/3.09	/3.09	/
unity3d.com	stats	7.01/	7.01/	7.01/	/
	config.uca.cloud	/0.01	/0.01	/0.01	/0.01
	api.uca.cloud	0.01/	0.01/	0.01/	0.01/
facebook.com	graph	0.11/3.97	0.11/3.97	0.11/3.97	0.11/3.97
	m	0.04/2.42	0.04/2.42	0.04/2.42	0.04/2.42
	www	0.36/1.20	0.36/1.20	0.36/1.20	0.36/1.20
amazonaws.com	s3	0.19/3.20	0.19/3.20	0.19/3.20	0.19/3.20
	prod-static-images.s3	0.21/	0.21/	0.21/	0.21/
	s3-us-west-1	0.08/0.15	0.08/0.15	0.08/0.15	0.08/0.15
tapjoyads.com	ws	0.01/4.46	0.01/4.46	0.01/4.46	/

of their web presence, we were also interested in each extension's ability to block smaller, less popular third-party trackers. Furthermore, we analyzed if there exists a trend of blocked third parties from the more popular to the less popular websites. To this end, we first extracted the number of distinct third-party domains included in the plain requests (no blocking extension installed) for each 10,000 Alexa rank interval. This dataset was split into 3 distinct sets containing third parties that were included on (2-20)/(20-200)/200-10000) distinct first-party pages. Therefore, the first set includes third parties with the smallest footprint (web presence), whereas the last set comprises third parties with the largest one. For each browser extension we then analyzed the number of the third-party domains that were not blocked. We consider a site as being, at least partially, blocked, if the inclusion count drops to half of the lower bound (e.g., less than 1/10/100 inclusions left respectively). Therefore, if a browser extension blocks more third-party domains in each respective set, it will consequently have a lower rate of third-party domains still included.

The results of this analysis are outlined in Figure 5. One conclusion we can draw from the results is that third-party domains with a larger footprint seem to be blocked more effectively by all extensions. Furthermore, we see that *Ghostery* has the best performance in blocking third parties with more than 20 inclusions. However, as the first plot shows, *uBlock* has a better performance on third parties included in less than 20 pages per Alexa rank interval. Our results indicate that smaller tracking companies are able to avoid attention from blocking tools and thus persist regardless of the presence of tracker-blocking extensions. An interesting side-effect is that in the very competitive and crowded sector of third-party tracking, tracker-blocking tools with incomplete coverage are indirectly "helping" smaller players by blocking their larger competitors.

Fingerprinters. In addition to measuring the effectiveness of browser extensions in blocking well-known third parties, we also investigated their ability to block stateless fingerprinting services. These services are able to identify users based on different attributes that are exposed through their browser, like available fonts or installed extensions. To quantify each extension's ability to block fingerprinting, we leveraged the previously detected fingerprinters found by Acar et al. [8] as well as the newly identified fingerprinters by Englehardt et al. [9]. Specifically, we utilized the regular expressions provided by the authors of FPDetective on Github [39] and the URI's provided in Englehardts' paper. The results we gathered by applying the rules to our dataset are shown in Table 5. Between the two studies it seems that recently more third parties rely on fingerprinting to identify users. A number of the fingerprinting services we detected were not blocked by any of our evaluated web browser extensions such as *MERCADOLIBRE*, *SiteBlackBox*, or *CDN.net*. Even though some of these services were identified by both studies as providers of fingerprinting scripts, it is unfortunate to see that they are not completely blocked by all browser extensions. For example, *CDN.net* was identified by FPDetective (i.e., three years ago) and again by Englehardt et al., and yet none of the extensions includes it in its rule set. Furthermore, we noticed that the numbers between OpenWPM and our crawl are distributed differently. As an example, we look at the first three scripts in canvas fingerprinting. While Englehart et al. identified *doubleverify.com/dvtp_src_internal23.js* and *doubleverify.com/dvtp_src_internal24.js* as the fingerprinters with most inclusions, both of them were not included in our dataset. However, as we checked for the regular expression *doubleverify.com/dvtp_src_internal.*.js* we were able to identify a similar number of inclusions, albeit with slightly different names. We assume that fingerprinters change the names of their scripts to evade overly strict rule sets. We also noticed that for some instances we observed more invocations of fingerprinting scripts with activated browser extensions compared to our vanilla (plain) browser instance. This divergence is likely a result of additional measures by websites to combat clickfraud in the absence of other third-party identifiers [6]. Finally, our analysis of popular Android applications showed that *ThreatMetrix* was included in 149 applications, i.e., 1.64% of our sample. We found that only the extensive DNS-based block list "Mother of all AD-BLOCKING" [44] effectively blocked this fingerprinting service.

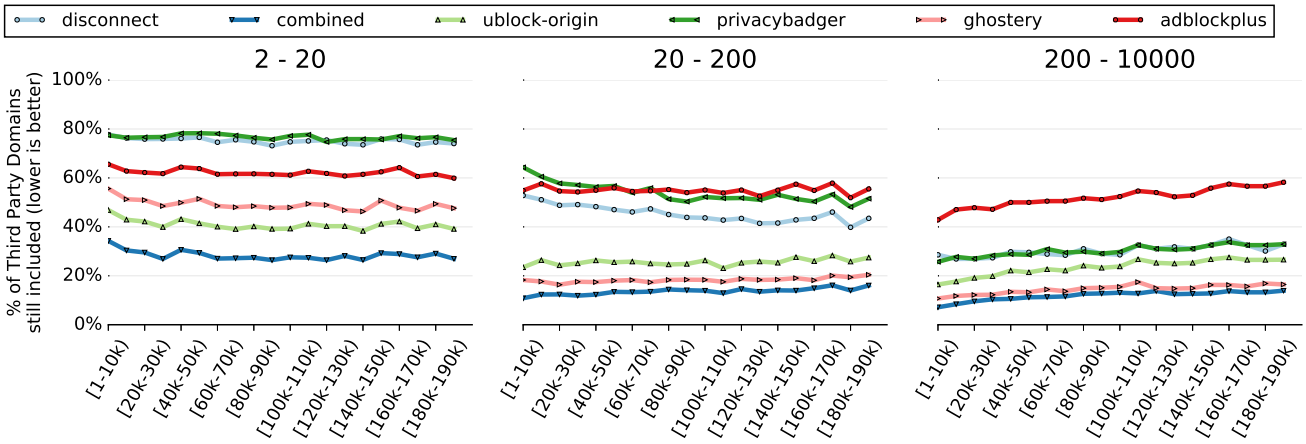

Figure 5. Sum of included third-party domains with 2-20/20-200/200-10000 inclusions which are not blocked by a specific browser extension in relation to the plain profile. In all graphs: the lower an extension is on the y-axis, the better (less third-parties remaining).

5.5. Overhead of Tracker Blocking

Blocking trackers with browser extensions comes at the cost of additional memory and CPU overhead for matching requests against their blocklists. This section discusses the overhead of the browser extensions we measured. These findings are especially important to mobile devices where mobile browsers are slowly opening up to extensions, but computing resources and battery life are still a limited commodity.

Overall, our evaluated browser extensions did not cause a significant CPU overhead. Two of the tested extensions even led to a reduction of the overall CPU usage of the tested web browser (*Disconnect* and *uBlock*). Figure 6 shows that browser extensions have a considerable impact on overall memory consumption. For this analysis, we first excluded the results from browser instances where we did not collect continuous samples of 30 accessed webpages, since browser instances were restarted, because one of the tasks (three webpages of one website) failed. This resulted in different sample subsets for the different browser extensions. We found that all browser extensions resulted in a higher initial memory consumption. After 30 webpages were accessed, the memory footprint varied depending on the used browser extension. We assume this happens, because the initial memory overhead is slowly amortized by the resource savings of blocking third-party trackers. Adblock Plus was responsible for a significant memory overhead, although none of the extensions resulted in less memory being used.

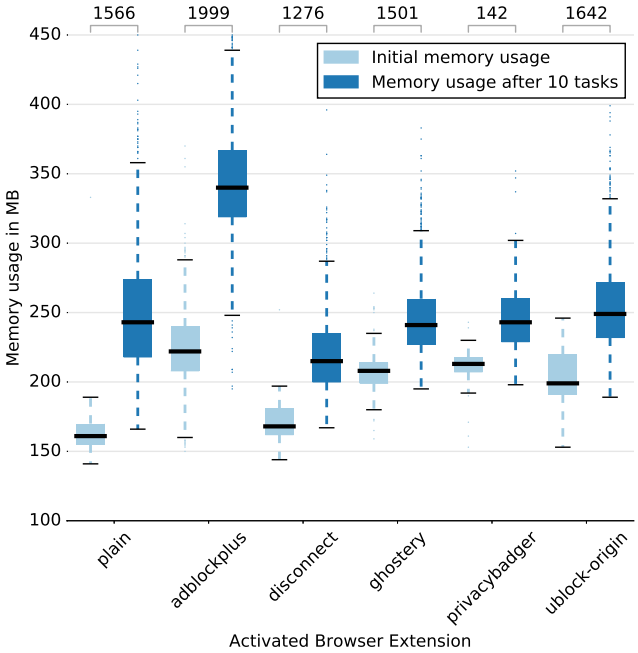

Figure 6. Initial memory usage and memory usage after requesting 30 web pages per browser extensions in MB. The numbers on the top show the amount of samples for each of the extensions.

6. Discussion

After presenting our large-scale analysis on the effectiveness of tracker-blocking tools on websites and mobile applications, we now discuss the implications of our findings for designing future tracker-blocking solutions.

6.1. Limitations

Our paper tackles the challenge of evaluating the effectiveness of tracker-blocking tools on a large scale. However, like any other large-scale study, our work has some limitations.

First, our results on third-party tracking only provide lower bounds, because our analysis does not account for content behind registration walls, since we cannot obtain accounts (paid and free) for thousands of websites. Second,

TABLE 5. NUMBER OF PAGES WITH DETECTED FINGERPRINTING SERVICES, LISTED WITHOUT ANY EXTENSIONS (PLAIN) AND PER BLOCKING EXTENSION. (SEE TABLE 3 FOR A DESCRIPTION OF THE TABLE HEADER).

	Web Dataset							Android			
	plain	abp	dc	gh	pb	ubo	c	plain	e	a	m
FP-Detective											
BlueCava	27	5	0	0	25	0	0	0	0	0	0
Myfreecams	0	0	0	0	0	0	0	0	0	0	0
Mindshare Tech.	1	1	1	1	1	0	0	0	0	0	0
AFK Media	2	2	2	2	1	2	1	0	0	0	0
CDN.net	15	14	17	17	14	16	11	0	0	0	0
ANALYTICSPROS	0	0	0	0	0	0	0	0	0	0	0
Anonymizer	0	0	0	0	0	0	0	0	0	0	0
AAMI	0	0	0	0	0	0	0	0	0	0	0
VIRWOX	1	1	1	1	1	1	1	0	0	0	0
ISINGLES	0	0	0	0	0	0	0	0	0	0	0
BBelements	0	0	0	0	0	0	0	0	0	0	0
Inside graph	16	17	16	0	15	0	0	0	0	0	0
PIANOMEDIA	0	0	0	0	0	0	0	0	0	0	0
ALIBABA	0	0	0	0	0	0	0	0	0	0	0
MERCADOLIBRE	4	5	5	5	4	5	3	0	0	0	0
LIGATUS	0	0	0	0	0	0	0	0	0	0	0
ThreatMetrix	39	39	39	1	37	0	0	149	149	149	0
IOVATION	98	97	97	1	97	4	1	0	0	0	0
MaxMind	14	13	13	14	12	1	1	1	1	1	0
Analytics-engine	0	0	0	0	0	0	0	0	0	0	0
Coinbase	0	0	0	0	0	0	0	0	0	0	0
SiteBlackBox	11	11	12	12	11	11	11	0	0	0	0
Perferencement	0	0	0	0	0	0	0	0	0	0	0
OpenWPM: Canvas Font Fingerprinting											
mathid.mathtag.com/device/id\.js	121	16	1	1	0	1	0	0	0	0	0
mathid.mathtag.com/d/i\.js	437	374	2	1	2	3	0	0	0	0	0
admicro1.vcmedia.vn/core/fipmin\.js	39	1	0	3	41	1	0	0	0	0	0
.*.online-metrix.net	39	39	39	1	37	0	0	149	149	149	0
pixel.infernotions.com/pixel/	6	6	6	1	7	1	1	0	0	0	0
api.twisto.cz/v2/proxy/test	0	0	0	0	0	0	0	0	0	0	0
go.lynxbroker.de/eat_session\.js	0	0	0	0	0	0	0	0	0	0	0
OpenWPM: Canvas Fingerprinting											
doubleverify.com/dvtp_src_internal.*\.js	4118	78	8	6	37	9	0	0	0	0	0
doubleverify.com/dvtp_src_internal24\.js	0	0	0	0	0	0	0	0	0	0	0
doubleverify.com/dvtp_src_internal23\.js	0	0	0	0	0	0	0	0	0	0	0
ap.lijit.com/sync	799	41	1	3	364	1	0	1	1	1	0
cdn.doubleverify.com/dvbs_src\.js	1781	26	4	2	51	4	0	0	0	0	0
rtbcdn.doubleverify.com/bsredirect5\.js	290	8	2	0	2	0	0	0	0	0	0
g.alicdn.com/alilog/mlog/aplus_v2\.js	116	121	129	131	48	0	0	0	0	0	0
static.audienceinsights.net/t\.js	39	17	27	25	30	0	0	0	0	0	0
static.boo-box.com/javascripts/embed\.js	21	0	21	0	21	0	0	0	0	0	0
admicro1.vcmedia.vn/core/fipmin\.js	39	1	0	3	41	1	0	0	0	0	0
c.imedia.cz/js/script\.js	45	35	45	0	43	38	0	0	0	0	0
ap.lijit.com/www/delivery/fp	826	27	296	3	349	1	0	1	0	1	0
www.lijit.com/delivery/fp	20	1	8	0	8	0	0	0	0	0	0
*amazonaws.com/af-bdaz/bquery\.js	40	0	35	0	32	0	0	0	0	0	0
.cloudfront.net/./platform.min\.js	23	24	23	3	25	15	2	0	0	0	0
voken.eyereturn.com/	41	0	1	0	13	0	0	0	0	0	0
*.hwcdn.net/fp/Scripts/PixelBundle\.js	2	1	1	1	1	0	0	0	0	0	0
static.fraudmetrix.cn/fm\.js	11	11	11	11	14	11	9	0	0	0	0
e.e701.net/cpc/js/common\.js	11	11	12	9	10	10	5	0	0	0	0
tags.bkrtx.com/js/bk-coretag\.js	631	391	134	0	449	6	0	10	10	10	0
dtt617kogtcso.cloudfront.net/sauce.min\.js	1	1	1	1	1	1	1	0	0	0	0
OpenWPM: WebRTC Local IP discovery											
cdn.augur.io/augur.min\.js	111	31	4	43	57	21	2	0	0	0	0
click.sabavision.com/*/jsEngine\.js	78	54	81	84	77	56	46	0	0	0	0
static.fraudmetrix.cn/fm\.js	11	11	11	11	14	11	9	0	0	0	0
.hwcdn.net/fp/Scripts/PixelBundle\.js	2	1	1	1	1	0	0	0	0	0	0
www.cdn-net.com/cc\.js	15	14	17	17	14	16	11	0	0	0	0
scripts.poll-maker.com/3012/scpolls\.js	3	3	3	3	4	3	3	0	0	0	0
static-hw.xvideos.com/vote/displayFlash\.js	6	7	9	9	7	10	6	0	0	0	0
g.alicdn.com/security/umscript/3.0.11/um\.js	0	0	0	0	0	0	0	0	0	0	0
load.instinctiveads.com/s/js/afp\.js	0	0	0	0	0	0	0	0	0	0	0
cdn4.forter.com/script\.js	4	4	4	2	5	0	0	0	0	0	0
socauth.privatbank.ua/cp/handler\.html	2	3	2	1	0	1	0	0	0	0	0
retailautomata.com/ralib/magento/raa\.js	0	0	0	0	0	0	0	0	0	0	0
live.activeconversion.com/ac\.js	0	0	0	0	0	0	0	0	0	0	0
*.ml.com/publish/ClientLoginUI/HTML/cc\.js	3	2	1	3	2	2	1	0	0	0	0
cdn.geocomply.com/101/gc-html5\.js	0	0	0	0	0	0	0	0	0	0	0
retailautomata.com/ralib/shopifynew/raa\.js	0	0	0	0	0	0	0	0	0	0	0
2nyan.org/animal/	2	2	2	2	2	2	2	0	0	0	0
pixel.infernotions.com/pixel/	6	6	6	1	7	1	1	0	0	0	0
167.88.10.122/ralib/magento/raa\.js	0	0	0	0	0	0	0	0	0	0	0

we used the default settings of all browser extensions, with the exception of Ghostery where we manually activated blocking of third-party trackers (the default mode of Ghostery is to only report the presence of trackers, but not to block them). Therefore, users must not misunderstand our experiments and arrive at the conclusion that merely installing Ghostery currently offers the best protection against trackers. Finally, the PrivacyBadger extension was limited to blocking trackers based on our training on the Alexa top 1,000 websites. As such, given more realistic workloads or usage spanning many days, PrivacyBadger could perform better (or worse) than it did in our measurements.

6.2. Future Tracking Defenses

Based on our findings we identify the following major challenges for future tracker-blocking browser extensions:

Social widgets are important. Despite the reach and impact of social widgets on the tracking of users, a number of existing browser extensions failed to effectively block Facebook's and Twitter's widgets. Future tracking defenses should focus on the creation of effective surrogates for common social widgets.

Creation of filter rules. We identified the need for research to automate or at least assist the laborious process of creating tracker-blocking filter rules. Previous research relied on the community-driven EasyList and EasyPrivacy filter rules [48], [49], [50] our findings suggest that the centralized rule sets by Ghostery or Disconnect might provide a better baseline for future research.

Closing blindspots. In addition to the varying effectiveness of different browser extensions regarding different stateful online trackers, all evaluated browser extensions failed to completely block well-known stateless fingerprinting services. We think that this is because of the opaque nature of fingerprinting which makes it harder for users to spot and hence report. Ideally, novel research into detecting stateless fingerprinters would automatically create blocking rules (since for some of the identified fingerprinters even after three years no filter rules exist). Finally, our results suggest that the proprietary filter rules of Ghostery should be complemented with the community-based rules from uBlock, to account for less popular third-party trackers.

Methodology for detecting broken websites. Our findings highlight an important research challenge for the automated creation of blocking rules: the risk of breaking websites. The heuristic creation of blocking rules with EFF's Privacy badger showed promising effectiveness, but also led to the highest number of unresponsive websites in our sample. Future research should therefore focus on methods to automatically detect whether a certain rule may break the functionality of websites.

Provide for content distribution networks. This ideal combination of filter rules would ultimately reduce third-party services to content distribution networks which cannot be blocked without breaking the functionality of websites. The growing usage of CDNs could ultimately thwart all existing tracker-blocking tools since CDN providers could deploy probabilistic stateless tracking based on the IP address and user agent of their users. A possible countermeasure for future tracker-blocking solutions would be the

inclusion of popular JS libraries/fonts to make requests to CDNs unnecessary. There is currently one browser extension for Mozilla Firefox which complements tracker-blocking browser extensions by locally providing popular CDN content [51].

Mobile in-app tracking. Mobile devices are often neglected in the discussion of third-party tracking protection, despite their growing usage. Currently, the most common browser extensions for blocking web trackers are available for Android (through Firefox Mobile) and iOS [15]. Unfortunately, tracking by mobile applications is harder to block. The rooting of mobile phones is outside the reach of everyday users, and therefore blocking can only be performed at a network level. Our results showed that AdAway/MoaAB (DNS-based blocking) and Adblock Plus for Android (proxy-based blocking) fail to significantly impact tracking by third parties on mobile applications.

6.3. Online Tracking and Security

In our large scale analysis, blocking extensions did not result in noticeable CPU overhead, and in the case of Disconnect it even led to decreased CPU usage. The majority of analyzed browser extensions, however, led to an increased memory footprint. Adblock Plus resulted in the biggest overhead which can be attributed to its use of cosmetic CSS-based filters which hide advertisements and the space they used to occupy, in addition to blocking them. Despite the memory overhead we measured, tracker-blocking has additional benefits for the security of users.

Third-party tracking and third-party content in general have been exploited as an attack vector in the past. The NSA used tracking identifiers to identify targets for further exploitation by passively analyzing unencrypted traffic en route to third-party tracking services. Our measurements showed that over 60% of third-party services did not use TLS to protect third-party requests and responses. Next to passive attacks abusing unprotected requests and responses, a web-wide over-reliance on specific third-party trackers can also be abused by active adversaries. For example, in a recent nation-state attack later dubbed "Great Cannon" [52], attackers replaced advertisements and analytics code loaded from *baidu.com* with malicious code which performed `DoS` attacks against specific targets. This example shows that popular third-party services can make very attractive targets for attackers. For example, based on our measurements, if attackers would be able to successfully attack *google-analytics.com*, they could push malicious code to approximately 70% of the top web pages.

7. Related Work

Measurement studies. To the best of our knowledge, there has been no study on blocking third-party trackers of a scope comparable to our work. Mayer and Mitchell [17] provide a survey describing how companies track users online and discuss current protection strategies. In addition, they introduced *FourthParty* [53], a web measurement tool to analyse third-party tracking. The authors used FourthParty to analyse the effectiveness of different blocking tools based on the Alexa Top 500. They found *Adblock Plus* with *EasyList* and *EasyPrivacy* to be the most effective browser extension. Balebako et al. [12] proposed a research methodology to analyze behavioral advertising based on web history and textual Google ads. They analyzed the built-in browser functionality, *Ghostery*, *Abine*, and the *Do not Track* header. They found that blocking tools were effective against behavioral profiling by Google text ads, because these tools completely removed the JavaScript code that generated them. Roesner et al. [3] analyzed the Top 500 websites, Non-Top 500, and the AOL testdata set in 2012 on existing trackers using a custom Firefox extension. Furthermore, they proposed a classification of different trackers according to their relationship with first-party websites and users and analyzed different browser-protection mechanisms. Reisman et al. [13] showed that the current state of third-party tracking enables surveillance and analyzed different blocking strategies. They found that Ghostery was the most effective tool for blocking trackers on the Alexa top 500 websites. All mentioned research has one key finding in common: dedicated tracker blockers significantly outperform other protection methods such as the disabling of third-party cookies, the usage of the Do-Not-Track header, and the setting of opt-out cookies. Our paper provides new insights in the arms race between third-party blockers and tracker blockers and demonstrates that the use of tracker blockers can have unexpected side-effects, such as the indirect assistance of smaller third-party trackers by blocking popular ones.

Stateless tracking. Motivated by the initial findings of Eckersley [19], a number of researchers further investigated stateless tracking and its implications. Yen et al. [54] performed a fingerprinting study similar to Eckersley's by analyzing logs of Bing and Hotmail. Interestingly, the authors found that a client's IP in combination with her user-agent string provided enough entropy to uniquely identify over 80% of the users. Nikiforakis et al. [6] described how fingerprinting works by analyzing the code of three browser-fingerprinting providers. Their work also showed that the spoofing of user-agents is an insufficient protection method that can cause more harm than good. Acar et al. [8] developed the *FPDetective* framework to detect web-based fingerprinters in the wild. They found that fingerprinting was used by over 400 domains in the Alexa Top 1 Million dataset. In a later study, the authors also investigated the usage of canvas-fingerprinting [55] in the wild as one more vector for uniquely identifying users across multiple websites [7]. The most extensive measurement on stateless tracking has been performed by Englehardt and Narayanan [9], including novel findings on the use of AudioContext fingerprinting. Our work leverages the findings of Englehardt and Narayanan as well as Acar et al. to shed light on the effectiveness of the state-of-the-art blocker tracking tools against stateless tracking on popular websites and mobile

apps. Our results showed that stateless tracking constitutes a serious blindspot of today's tracker-blocking tools.

Tracking defense strategies. A number of defense strategies has been proposed in the past. Guha et al. [56] proposed Privad which acts as a privacy-preserving dealer for advertising. The approach by Guha et al. tries to find a balance between privacy and still showing relevant ads to users. To the best of our knowledge, no systems comparable to Privad are used by advertisement providers. Recent proposals like the TrackingFree browser by Pan et al. [57] rely on separated identities (browser principals) for different websites in order to hinder tracking by third parties. A similar approach has been proposed by Torres et al. [58] where a custom browser extension enforces separate web identities per website [58]. Privaricator [59] uses a modified stock web browser to fake browser fingerprints. These proposals offer promising methods to hinder stateful and stateless tracking, but they all rely on browser vendors adopting their technologies. Tracker-blocking browser extensions thus offer the best protection strategy against online tracking and also have positive side-effects, such as protecting users from malvertising and active URL hijacking attacks [52]. Cranor [10] showed that these tools are plagued by usability issues and proposed improvements. The usability survey by Leon et al. [60] furthermore highlighted that only one out of five participants was able to enable the optional blocking feature of Ghostery, the most effective tool in our measurement study.

Finally, given the effectiveness of tracker-blocking tools, the research community focuses on improving the underlying tracker-blocking rules. Previous research suggests to leverage machine learning for complementing existing rule sets. Bau et al. [48] proposed to use supervised machine learning to detect tracking by third-party domains. Bhagavatula et al. [49] evaluated different machine learning algorithms and found that the k-nearest neighbor algorithm outperformed the accuracy of other classifiers. They used EasyList as a baseline and analyzed if they could correctly predict URLs included in this popular ad-blocking list. Gugelman et al. [50] used a naive Bayes classifier to detect privacy-intrusive services based on statistical HTTP traffic features. It is interesting to note that the research community bases their experiments on the EasyList and EasyPrivacy rule sets with which they train their detection classifiers. Our results suggest that other rule sets, such as those of Ghostery's or Disconnect might provide a better benchmark for machine learning experiments.

8. Conclusion

In this paper we conducted a large-scale analysis of the widespread practice of online tracking by third-party services. Third-party tracking has serious implications for the privacy and security of users, and we provide insights into the effectiveness of current tracker-blocking tools. Our results are based on the analysis of over 100,000 websites in combination with the state-of-the-art tracker blocking tools. Our findings suggest that some browser extensions can effectively block the majority of stateful third-party trackers, although still having blind spots regarding blocking stateless fingerprinting scripts and smaller third-party trackers. We furthermore showed that over 60% of third-party tracking services communicate in plain text. Finally, our paper discussed the unique challenges of effectively blocking third-party trackers on mobile devices.

Overall, the contributions of this paper advance the field of web privacy by providing not only the largest study on the effectiveness of tracker-blocking tools on websites to date, but furthermore highlighting the most pressing challenges for mitigating online tracking.

Acknowledgements

This work has been carried out within the scope of "u'smile", the Josef Ressel Center for User-Friendly Secure Mobile Environments, funded by the Christian Doppler Gesellschaft, A1 Telekom Austria AG, Drei-Banken-EDV GmbH, LG Nexera Business Solutions AG, NXP Semiconductors Austria GmbH, and Österreichische Staatsdruckerei GmbH. Furthermore, this work has been supported by the Austrian Research Promotion Agency under grant 853264 (PriSAd), a joint project of Nimbusec GmbH and the St. Pölten University of Applied Sciences. For Stony Brook University, this research was supported by the National Science Foundation (NSF) under grants CNS-1527086 and CNS-1617593.

References

[1] The Guardian, ""tor stinks" presentation," 2013, http://www.theguardian.com/world/interactive/2013/oct/04/tor-stinks-nsa-presentation-document.

[2] G. Greenwald, "Xkeyscore: Nsa tool collects' nearly everything a user does on the internet'," *The Guardian*, vol. 31, 2013.

[3] F. Roesner, T. Kohno, and D. Wetherall, "Detecting and defending against third-party tracking on the web," in *NSDI*. USENIX Association, 2012.

[4] B. Krishnamurthy and C. E. Wills, "Generating a privacy footprint on the internet," in *ACM Internet Measurement Conference*. ACM, 2006, pp. 65–70.

[5] C. J. Hoofnagle, A. Soltani, N. Good, D. J. Wambach, and M. Ayenson, "Behavioral advertising: The offer you cannot refuse," *Harvard Law & Policy Review*, 2012.

[6] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless monster: Exploring the ecosystem of web-based device fingerprinting," in *Security and privacy (SP), 2013 IEEE symposium on*. IEEE, 2013, pp. 541–555.

[7] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The web never forgets: Persistent tracking mechanisms in the wild," *ACM CCS'14*, 2014.

[8] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel, "Fpdetective: Dusting the web for fingerprinters," in *ACM CCS'13*. ACM, 2013, pp. 1129–1140.

[9] S. Englehardt and A. Narayanan, "Online tracking: A 1-million-site measurement and analysis Draft: July 11th, 2016," Jul. 2016, [Technical Report]. [Online]. Available: http://randomwalker.info/publications/OpenWPM_1_million_site_tracking_measurement.pdf

[10] L. F. Cranor, "Can users control online behavioral advertising effectively?" *Security & Privacy, IEEE*, vol. 10, no. 2, pp. 93–96, 2012.

[11] J. Mayer, "Tracking the trackers: Early results," *The Center for Internet and Societys Blog*, 2011.

[12] R. Balebako, P. Leon, R. Shay, B. Ur, Y. Wang, and L. Cranor, "Measuring the effectiveness of privacy tools for limiting behavioral advertising," in *W2SP*, 2012.

[13] D. Reisman, S. Englehardt, C. Eubank, P. Zimmerman, and A. Narayanan, "Cookies that give you away: Evaluating the surveillance implications of web tracking," in *WWW*, 2014.

[14] G. Kontaxis and M. Chew, "Tracking protection in firefox for privacy and performance," *W2SP*, 2014.

[15] Engadet, "iOS 9's web browser can block annoying ads," 2015. [Online]. Available: http://www.engadget.com/2015/06/10/safari-in-ios-9-can-block-ads/

[16] D. Malandrino, A. Petta, V. Scarano, L. Serra, R. Spinelli, and B. Krishnamurthy, "Privacy awareness about information leakage: who knows what about me?" in *12th ACM WPES workshop*. ACM, 2013, pp. 279–284.

[17] J. R. Mayer and J. C. Mitchell, "Third-party web tracking: Policy and technology," in *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012, pp. 413–427.

[18] S. Kamkar, "Evercookie-virtually irrevocable persistent cookies," *His Blog*, vol. 9, 2010.

[19] P. Eckersley, "How unique is your web browser?" in *Privacy Enhancing Technologies*. Springer, 2010, pp. 1–18.

[20] B. Krishnamurthy, K. Naryshkin, and C. Wills, "Privacy leakage vs. protection measures: the growing disconnect," *W2SP*, vol. 2, pp. 1–10, 2011.

[21] R. Stevens, C. Gibler, J. Crussell, J. Erickson, and H. Chen, "Investigating user privacy in android ad libraries," in *Workshop on Mobile Security Technologies (MoST)*. Citeseer, 2012.

[22] M. C. Grace, W. Zhou, X. Jiang, and A.-R. Sadeghi, "Unsafe exposure analysis of mobile in-app advertisements," in *WISEC*. ACM, 2012, pp. 101–112.

[23] T. Book, A. Pridgen, and D. S. Wallach, "Longitudinal analysis of android ad library permissions," *arXiv preprint arXiv:1303.0857*, 2013.

[24] J. Mayer, "The Turn-Verizon Zombie Cookie," 2015, http://webpolicy.org/2015/01/14/turn-verizon-zombie-cookie/.

[25] Ars Technica, "How a banner ad for H&R Block appeared on apple.com without Apple's OK |," 4 2013, http://arstechnica.com/tech-policy/2013/04/how-a-banner-ad-for-hs-ok/.

[26] MVPS, "Blocking unwanted connections with a hosts file," 2015, http://winhelp2002.mvps.org/hosts.htm.

[27] P. Lowe, "Yoyo hosts file," 2015, http://pgl.yoyo.org/as/.

[28] D. Schürmann, D. Mošenkovs, and 0-kaladin, "Adaway," https://www.adaway.org, 2016.

[29] F. Keil, D. Schmidt, H. Burgiss, L. Rian, and R. Rosenfeld, "Privoxy," 2015, http://www.privoxy.org/.

[30] T. Register, "Ssl-busting adware: Us cyber-plod open fire on comodo's privdog," 2015, http://www.theregister.co.uk/2015/02/24/comodo_ssl_privdog/.

[31] C. Evans, C. Palmer, and R. Sleevi, "Public key pinning extension for http," http://www.rfc-editor.org/rfc/rfc7469.txt, 2015.

[32] M. Kranch and J. Bonneau, "Upgrading https in mid-air: An empirical study of strict transport security and key pinning." in *NDSS*, 2015.

[33] Adblock Plus, "Mercurial repositories index," https://hg.adblockplus.org/.

[34] W. Palant, "Allowing acceptable ads in adblock plus," 12 2011, https://adblockplus.org/development-builds/allowing-acceptable-ads-in-adblock-plus.

[35] Ars Technica, "Over 300 businesses now whitelisted on adblock plus, 10% pay to play," 2015, http://arstechnica.com/business/2015/02/over-300-businesses-now-whitelisted-on-adblock-plus-10-pay-to-play/.

[36] S. Englehardt, C. Eubank, P. Zimmerman, D. Reisman, and A. Narayanan, "Openwpm: An automated platform for web privacy measurement," 2015.

[37] A. Cortesi, "mitmproxy," https://mitmproxy.org/, 2015.

[38] M. Foundation, "Public suffix list," https://publicsuffix.org/, 2015.

[39] G. Acar, "Fpdetective regular expressions," https://github.com/fpdetective/fpdetective/blob/master/src/crawler/fp_regex.py, 2016.

[40] N. Viennot, E. Garcia, and J. Nieh, "A measurement study of google play," in *SIGMETRICS*. ACM, 2014, pp. 221–233.

[41] A. Desnos, "Androguard: Reverse engineering, malware and goodware analysis of android applications... and more (ninja!)," https://github.com/androguard/androguard, 2015.

[42] Google, "Admob mediation network," https://developers.google.com/admob/android/mediation-networks, 2015.

[43] Genymobile, "Genymotion," https://www.genymotion.com, 2015.

[44] BSDgeek_Jake, "Moaab: Mother of all ad-blocking," http://forum.xda-developers.com/showthread.php?t=1916098, 2015.

[45] A. Novikov, "Adblock plus 1.0 for android released," https://adblockplus.org/releases/adblock-plus-10-for-android-released, 2012.

[46] M. Falahrastegar, H. Haddadi, S. Uhlig, and R. Mortier, "Anatomy of the third-party web tracking ecosystem," *arXiv preprint arXiv:1409.1066*, 2014.

[47] A. Lutz, "Not blocking google analytics?" https://github.com/EFForg/privacybadgerfirefox/issues/298, 2015.

[48] J. Bau, J. Mayer, H. Paskov, and J. C. Mitchell, "A promising direction for web tracking countermeasures," *W2SP*, 2013.

[49] S. Bhagavatula, C. Dunn, C. Kanich, M. Gupta, and B. Ziebart, "Leveraging machine learning to improve unwanted resource filtering," in *AISec*. ACM, 2014, pp. 95–102.

[50] D. Gugelmann, M. Happe, B. Ager, and V. Lenders, "An automated approach for complementing ad blockers blacklists," *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 282–298, 2015.

[51] T. Rientjes, "Decentraleyes," 2016, https://decentraleyes.org.

[52] B. Marczak, N. Weaver, J. Dalek, R. Ensafi, D. Fifield, S. McKune, A. Rey, J. Scott-Railton, R. Deibert, and V. Paxson, "An analysis of china's "great cannon"," in *5th USENIX FOCI Workshop*. Washington, D.C.: USENIX Association, Aug. 2015.

[53] J. Mayer, "Fourthparty web measurement platform," http://fourthparty.info/, 2015.

[54] T.-F. Yen, Y. Xie, F. Yu, R. P. Yu, and M. Abadi, "Host fingerprinting and tracking on the web: Privacy and security implications." in *NDSS*, 2012.

[55] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in html5," in *Web 2.0 Workshop on Security and Privacy (W2SP)*, 2012.

[56] S. Guha, B. Cheng, and P. Francis, "Privad: Practical privacy in online advertising." in *NSDI*, 2011.

[57] X. Pan, Y. Cao, and Y. Chen, "I do not know what you visited last summer: Protecting users from third-party web tracking with trackingfree browser," in *NDSS*, 2015.

[58] H. J. Christof Torres and S. Mauw, "Fp-block: usable web privacy by controlling browser fingerprinting," in *ESORICS*, 2015.

[59] N. Nikiforakis, W. Joosen, and B. Livshits, "Privaricator: Deceiving fingerprinters with little white lies," in *WWW*. International World Wide Web Conferences Steering Committee, 2015, pp. 820–830.

[60] P. Leon, B. Ur, R. Shay, Y. Wang, R. Balebako, and L. Cranor, "Why johnny can't opt out: a usability evaluation of tools to limit online behavioral advertising," in *SIGCHI*. ACM, 2012, pp. 589–598.

Appendix A: Impact of Tracker Blocker Extensions on Major Third-Party Services

TABLE 6. THIS TABLES OUTLINES COMMON THIRD PARTIES WE DETECTED IN OUR SAMPLE OF 123,876 WEBSITES OUT OF THE ALEXA TOP 200,000. THE NUMBERS ACCOUNT FOR THE PERCENTAGE OF INCLUSION IN DIFFERENT WEBSITES WITH RESPECT TO THE TOTAL SAMPLE (HTTP/HTTPS).

		plain *http/https*	adblockplus *http/https*	disconnect *http/https*	ghostery *http/https*	privacybadger *http/https*	ublock-origin *http/https*	combined *http/https*
google-analytics.com	www	13.2/53.6	19.2/47.4	1.4/0.4	0.9/	54.6/13.7	/	/
	ssl	/6.9	/6.8	/	/	/5.7	/	/
		/	/	/	/	/	/	/
doubleclick.net	stats.g	/35.4	0.6/34.6	/	/	/	/	/
	googleads.g	3.3/31.9	1.5/13.7	/	/	/1.4	/	/
	cm.g	13.3/25.9	8.0/3.8	1.7/0.8	/	0.9/	/	/
google.com	www	13.8/41.1	7.7/24.3	6.9/14.6	4.9/13.1	6.8/14.0	5.6/14.0	5.0/10.2
	apis	/14.4	0.5/14.3	/8.5	/1.3	/	/14.1	/
	accounts	/10.1	/9.9	/3.8	/1.3	/	/9.8	/
gstatic.com	fonts	21.2/24.3	20.8/21.9	21.7/19.0	21.0/17.5	20.8/15.5	21.4/22.5	18.2/15.5
	www	4.0/17.3	1.1/4.0	1.4/4.0	0.6/2.9	1.4/3.7	0.6/3.9	0.6/2.1
	ssl	0.5/10.4	0.5/10.2	0.5/4.0	/1.5	/0.5	0.5/10.1	/0.5
googleapis.com	fonts	23.8/15.9	23.6/12.9	24.2/13.5	23.5/13.3	23.6/12.6	23.9/13.2	20.9/12.3
	ajax	16.2/10.3	15.5/9.6	16.2/9.9	15.4/9.6	16.0/9.4	15.5/9.6	13.3/8.3
	maps	1.8/2.4	1.8/2.4	1.8/2.6	1.9/2.6	1.7/2.3	1.8/2.5	1.3/1.8
facebook.com	www	1.4/37.9	1.7/35.7	/0.7	/1.0	/	1.0/22.8	/
	staticxx	2.8/22.7	4.3/22.5	/	/	0.8/0.8	2.9/22.7	/
	graph	1.9/2.0	1.9/1.8	/0.5	/	0.9/1.1	1.0/1.7	/
googlesyndication.com	pagead2	27.5/29.4	0.7/0.7	14.6/1.1	/	16.7/16.1	/	/
	tpc	4.5/20.1	/0.5	/	/	/	/	/
	video-ad-stats	/	/	/	/	/	/	/
facebook.net	connect	8.0/24.1	10.8/20.5	1.8/0.6	/	/	9.5/18.6	/
	www.connect	/	/	/	/	/	/	/
googleadservices.com	www	10.2/6.2	9.9/5.5	5.7/2.4	/	9.2/4.9	/	/
	partner	9.4/0.9	8.1/0.8	0.6/	/	8.8/0.8	/	/
	pagead2	/	/	/	/	/	/	/
twitter.com	platform	13.8/13.6	13.8/13.5	3.5/1.0	/	12.5/11.8	12.6/12.8	/
	syndication	/11.7	/11.5	/	/	/	/5.0	/
	analytics	/5.9	/2.6	/	/	/1.7	/	/
fbcdn.net	static.xx	/19.0	/18.8	/	/0.4	/	/18.5	/
	scontent.xx	/8.9	/8.7	/0.5	/	/0.5	/9.0	/
	external.xx	/1.0	/1.0	/	/	/	/1.1	/
adnxs.com	ib	14.8/4.1	7.2/1.1	1.2/0.4	/	5.7/0.8	/	/
	secure	0.4/4.7	/2.2	/	/	/2.5	/	/
	acdn	1.6/	/	/	/	0.6/	/	/
cloudfront.net	d5nxst8fruw4z	/2.8	/2.7	/2.8	/	/2.6	/	/
	d31qbv1cthcecs	/2.7	/2.6	/2.7	/	/2.5	/	/
	dnn506yrbagrg	1.5/0.4	1.5/0.4	1.5/0.5	/	1.5/0.4	/	/
yahoo.com	ads	7.5/3.2	/	/	/	/	/	/
	pr-bh.ybp	3.2/1.9	/	/	/	/	/	/
	cms.analytics	2.4/	1.7/	/	/	/	/	/
googletagmanager.com	www	9.9/4.5	9.8/4.5	9.9/4.6	/	9.7/4.3	/	/
		/	/	/	/	/	/	/
addthis.com	m	9.8/1.3	9.7/1.3	/	/	/	/	/
	s7	6.3/1.0	6.3/1.1	5.6/1.0	/	/	/	/
	su	5.0/0.4	3.8/	/	/	/	/	/
amazonaws.com	s3	2.2/2.1	2.0/2.0	2.1/2.0	1.7/1.5	2.1/1.8	1.6/1.6	1.3/1.1
	load.s3	2.7/1.0	1.3/	/	/	/	/	/
	cloudfront-labs	2.4/	2.3/	2.4/	/	2.3/	/	/
scorecardresearch.com	b	9.7/	7.3/	0.5/	/	/	/	/
	sb	/3.3	/1.2	/	/	/	/	/
	sa	/0.9	/	/	/	/	/	/
mathtag.com	sync	7.3/2.0	3.2/0.7	/	/	/	/	/
	pixel	4.9/0.7	2.5/	/	/	/	/	/
	tags	1.2/2.4	/	/	/	/	/	/
rlcdn.com	idsync	9.9/2.4	5.3/0.7	0.6/	/	/	/	/
	rc	1.6/	1.3/	/	/	/	/	/
	ei	/	/	/	/	/	/	/
2mdn.net	s0	1.9/9.2	/	/	/	/	/	/
	s1	/4.4	/	/	/	/	/	/
	s0qa	/0.9	/	/	/	/	/	/
adsrvr.org	match	9.0/2.2	/	/	/	/	/	/
	insight	0.7/0.7	/	/	/	/	/	/
	usw-lax	0.6/	/	/	/	/	/	/
openx.net	us-u	7.4/4.0	/	/	/	/	/	/
	us-ads	1.1/	/	/	/	/	/	/
	u	0.9/	/	/	/	/	/	/
bluekai.com	tags	9.2/3.3	4.6/	/	/	0.5/	/	/
	stags	/1.3	/0.5	/	/	/	/	/
	analytics	/	/	/	/	/	/	/
rubiconproject.com	pixel	6.6/4.7	/	/0.4	/	/	/	/
	optimized-by	1.8/	/	/	/	/	/	/
	ads	1.7/	/	/	/	/	/	/
googletagservices.com	www	9.0/2.4	7.6/1.0	0.5/	/	8.4/1.1	/	/
cloudflare.com	cdnjs	3.3/4.3	3.1/2.3	3.3/2.4	3.2/2.4	3.2/2.2	3.2/2.3	2.8/2.0
	ajax	2.3/	2.2/	2.3/0.4	2.3/	2.3/	2.2/	2.0/
	www	/	/	/	/	/	/	/
advertising.com	sync.adaptv	4.2/1.4	/	/	/	/	/	/
	pixel	2.8/0.5	/	/	/	/	/	/
	cas.pxl.ace	1.4/	/	/	/	/	/	/
bidswitch.net	x	7.3/3.2	1.3/1.0	0.8/0.6	/	/	/	/
	useast-aws2	/	/	/	/	/	/	/
	us-east	/	/	/	/	/	/	/