How Tracking Companies Circumvented Ad Blockers Using WebSockets

Muhammad Ahmad Bashir, Sajjad Arshad, Engin Kirda, William Robertson, Christo Wilson
Northeastern University
Boston, MA
{ahmad,arshad,ek,wkr,cbw}@ccs.neu.edu

ABSTRACT

In this study of 100,000 websites, we document how Advertising and Analytics (*A&A*) companies have used WebSockets to bypass ad blocking, exfiltrate user tracking data, and deliver advertisements. Specifically, our measurements investigate how a long-standing bug in Chrome's (the world's most popular browser) `chrome.webRequest` API prevented blocking extensions from being able to interpose on WebSocket connections. We conducted large-scale crawls of top publishers before and after this bug was patched in April 2017 to examine which A&A companies were using WebSockets, what information was being transferred, and whether companies altered their behavior after the patch. We find that a small but persistent group of A&A companies use WebSockets, and that several of them engaged in troubling behavior, such as browser fingerprinting, exfiltrating the DOM, and serving advertisements, that would have circumvented blocking due to the Chrome bug.

ACM Reference Format:
Muhammad Ahmad Bashir, Sajjad Arshad, Engin Kirda, William Robertson, Christo Wilson. 2018. How Tracking Companies Circumvented Ad Blockers Using WebSockets. In *2018 Internet Measurement Conference (IMC '18), October 31-November 2, 2018, Boston, MA, USA.* ACM, New York, NY, USA, 7 pages. `https://doi.org/10.1145/3278532.3278573`

1 INTRODUCTION

The use of techniques to block online ads and prevent tracking has proliferated in recent years. Measurement studies estimate that Adblock Plus is used by roughly 16–37% of web users [37, 49], and numerous other extensions like Ghostery, Disconnect, Privacy Badger, and uBlock Origin have devoted user bases.

In response to the proliferation of blocking and privacy tools, online Advertising and Analytics (A&A) companies have fought back in a variety of ways. This includes industry self-regulation such as the Ad Choices initiative [4], as well as technological mechanisms like anti-ad blocking scripts [41, 44]. Most alarmingly, some companies have attempted to circumvent privacy tools, with the most infamous case being Google's evasion of Safari's third-party cookie blocking policy, which resulted in a $22.5M settlement with the FTC [15]. Franken et al. [22] developed a principled set of tests to assess whether tracking countermeasures in browsers and extensions could be evaded, and found many flaws that could be leveraged by A&A companies (but were not at the time, circa 2018).

In August 2016, privacy conscious users began to notice ads appearing on specific websites on Chrome, despite the use of ad blocking extensions [27, 50]. Online sleuths determined that (1) these ads were being downloaded via WebSockets because (2) a long-standing bug in the `chrome.webRequest` API in Chromium [29] allowed WebSocket connections to bypass ad blocking extensions. We refer to this issue as the webRequest Bug (WRB). Google patched the WRB in Chrome 58, released on April 19, 2017 [48].

In this paper, we study the behavior of A&A companies with respect to the WRB. Prior to April 19, 2017, there existed a five year window in which blocking extensions in Chrome (the world's most popular web browser [56]) could be circumvented through the use of WebSockets. We ask the following questions: which A&A companies, if any, decided to leverage this bug? Similarly, after the release of Chrome 58, did A&A companies continue to use WebSockets, or did they revert to HTTP/S? These questions are important, as they speak to the lengths that A&A companies are willing to go to track users and monetize impressions.

To answer these questions, we performed four crawls of the top Alexa websites: two just prior to the release of Chrome 58, and two after. Our crawls were conducted using stock Chrome coupled with custom instrumentation to record the *inclusion tree* of resources within each webpage [5, 8, 35] (see § 3 for details).

Using this data, we make the following key findings:

- Although we find that WebSocket usage is rare overall (~2% of publishers), 65–72% of WebSockets are related to tracking and advertising in some way. Furthermore, we find that A&A sockets are more prevalent on Alexa top-10K publishers as compared to less popular publishers.
- We observe 94 A&A domains initiating and 20 A&A domains receiving WebSocket connections, including some of the largest players in the advertising ecosystem (e.g., DoubleClick and Facebook).
- We find that the overall frequency of WebSockets used by A&A domains did not change after the release of Chrome 58, although the number of unique WebSocket initiators dropped from 75 to 23, as major ad networks (e.g., DoubleClick) discontinued their use.
- We find sensitive information being sent over WebSockets to A&A companies. *33across* collected browser fingerprints [2, 19, 31, 43, 51], while *Hotjar*, *LuckyOrange*, and *TruConversion* collected the entire DOM, which can contain sensitive information such as search queries, unsent messages, etc., within the given webpage. *Lockerdome* was using WebSockets to serve URLs to ads. These results highlight that

ACM ISBN 978-1-4503-5619-0/18/10...$15.00
`https://doi.org/10.1145/3278532.3278573`

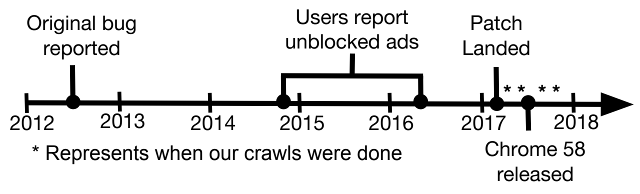


Figure 1: Timeline for the WRB and our four crawls.

the WRB did enable A&A companies to circumvent blocking extensions in ways that users may find objectionable.

2 BACKGROUND

We begin by providing an overview of WebSockets, the `webRequest` API, and a brief timeline of the WRB.

2.1 WebSockets

The WebSocket protocol, standardized by RFC 6455 in 2011, gave JavaScript developers access to a bidirectional, socket-like network interface, in which client-side JavaScript can open a WebSocket connection to a server. This protocol enables developers to create web applications that receive real-time information or “pushed” messages from the server-side, without wasting bytes or incurring latency due to the constant construction of new TCP connections.

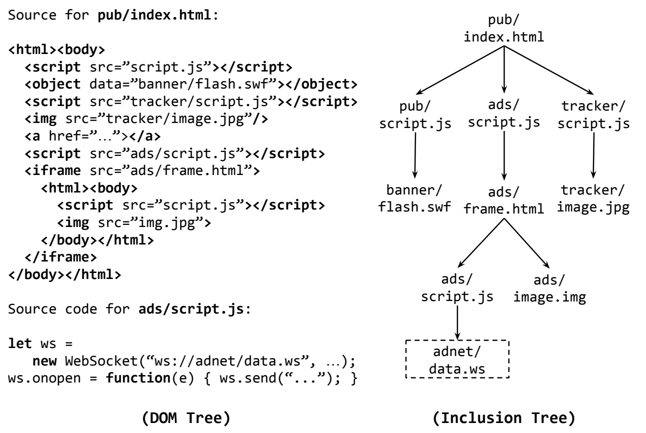
2.2 webRequest API

As of 2017, major browser vendors like Firefox and Edge support the Chrome extension API. One of its key capabilities is the `chrome.webRequest` API, which allows extensions to inspect, modify, and even drop outgoing network requests. The `chrome.webRequest.onBeforeRequest` callback is often used by ad blockers and privacy preserving tools to filter undesirable outgoing network requests [29].

2.3 The Rise and Fall of a Bug

In May 2012, users created a bug report in the Chromium issue tracker after observing that WebSocket connections did not trigger the `chrome.webRequest.onBeforeRequest` callback [29]. We refer to this as the webRequest Bug (WRB).

The WRB languished unpatched for four years. In late 2014, AdBlock Plus users began to report that unblockable ads were appearing on specific webpages, but only in Chrome [3]. By mid-2016, EasyList and uBlock Origin users were also observing unblockable ads [27, 50]. The users investigated and determined that the ads were being loaded via WebSockets, i.e., the WRB was being leveraged by some ad networks to circumvent blockers. Blocking extensions implemented complicated workarounds to mitigate the WRB in the absence of a permanent bugfix [3, 26]. In November 2016, Pornhub was caught circumventing ad blocking extensions using WebSockets [11, 38]. The WRB was finally patched in Chrome 58, released in April 19, 2017 [48]. Figure 1 shows the timeline of key events related to the WRB.



Source for **pub/index.html**:

```
<html><body>
  <script src=”script.js”></script>
  <object data=”banner/flash.swf”></object>
  <script src=”tracker/script.js”></script>
  <img src=”tracker/image.jpg”/>
  <a href=”...”></a>
  <script src=”ads/script.js”></script>
  <iframe src=”ads/frame.html”>
    <html><body>
      <script src=”script.js”></script>
      <img src=”img.jpg”>
    </body></html>
  </iframe>
</body></html>
```

Source code for **ads/script.js**:

```
let ws =
   new WebSocket(“ws://adnet/data.ws”, ...);
ws.onopen = function(e) { ws.send(“...”); }
```

Figure 2: Sample DOM tree with corresponding *inclusion tree.* Note how a WebSocket request becomes a child to the requesting JavaScript resource.

3 METHODOLOGY

The goal of our study is to analyze WebSockets usage on the web, and to determine whether A&A companies were using them to bypass ad blockers. In this section, we outline our data collection methodology. We also describe the inclusion trees produced by our crawler, and explain how we use these to attribute WebSockets.

3.1 Inclusion Tree

To determine which A&A companies were using WebSockets to circumvent blockers, we are not only interested in determining the existence of a WebSocket on a webpage, but also figuring out which parties established the socket in the first place. Prior studies have shown that relying on HTTP requests to determine resource inclusions can be misleading due to dynamic code (e.g., JavaScript, Flash etc.) from third parties [8]. This occurs because the *Referer* header is set to the first-party domain, even if the resource making the request originated from a third-party. Furthermore, using the Document Object Model APIs (a.k.a. the DOM; programming interface for HTML and XML documents [23]) to capture resource inclusions also does not work because it encodes *syntactic* structures rather than *semantic* relationships between resource inclusions.

To solve this problem we use *inclusion trees*, introduced by Arshad et al. [5]. Inclusion trees capture the semantic relationship between resource inclusions in websites. Figure 2 shows a sample DOM tree and its corresponding inclusion tree. We capture inclusion trees from Chrome by leveraging the Chrome Debugging Protocol [14]. Specifically, to capture the inclusion relationships within Chrome using the `Debugger` domain, we track JavaScript by collecting the `scriptParsed` events, which are triggered by the execution of inline and remote scripts. We observe further resource requests via the `requestWillBeSent` and `responseReceived` events in the `Network` domain. Using these two events, we can capture most of the dynamic inclusion chains. To capture the `iframe` inclusions, we collect `frameNavigated` events in the `Page` domain.

3.2 WebSocket Detection and Labeling

A main distinguishing feature of our tool from previous work [5, 8, 35] is its ability to detect WebSocket requests initiated by JavaScript. In our implementation, we treat WebSockets as child nodes of the JavaScript node responsible for initiating them. Figure 2 shows how `adnet/data.ws` becomes the child of `ads/script.js`. To identify WebSocket requests, we capture a number of events in the `Network` domain: `webSocketCreated`, `webSocketWillSendHandshakeRequest`, and `webSocketHandshakeResponseReceived` for socket initiation; `webSocketFrameSent` and `webSocketFrameReceived` for data collection; and `webSocketClosed` for socket termination.

Detecting A&A Resources. To determine whether a socket was initiated by scripts or objects that originated from A&A domains, we first derive a set of A&A domains from the inclusion chains provided by Bashir et al. [8]. Each resource in [8] is tagged as A&A or non-A&A using the EasyList and EasyPrivacy rule lists; from this dataset, we extract a set of all 2^{nd}-level domains D (e.g., 2^{nd}-level domain for both `x.doubleclick.net` and `y.doubleclick.net` will be `doubleclick.net`). Let $a(d)$ and $n(d)$ be the number of times a given 2^{nd}-level domain $d \in D$ was labeled as A&A and non-A&A, respectively. We construct our final A&A set D' containing all $d \in D$ where $a(d) \geq 0.1 * n(d)$, i.e., we filter out 2^{nd}-level domains that are labeled as A&A less than 10% of the time to eliminate false positives.

The one exception to this process was Amazon's Cloudfront CDN, which we observed hosting ad-related scripts and images. We found 13 unique fully-qualified Cloudfront domains (e.g., `dkpklk99llpj0.cloudfront.net`) that immediately preceded or succeeded A&A domains in our dataset. We manually mapped each of these Cloudfront domains to the specific A&A company hosting content there by examining the order of resource loads in the corresponding inclusion chains. In most cases this mapping was trivial, since we observed a one-to-one relationship between JavaScript from a specific A&A company and a specific Cloudfront subdomain (e.g., LuckyOrange and `d10lpsik1i8c69.cloudfront.net`).

To detect WebSockets that were initiated by A&A resources, we descend the branch of the inclusion tree that includes the socket. If the domains of any of the parent resources are present in D', we consider the socket to be included by an A&A resource. We refer to such sockets as *A&A sockets*.

3.3 Data Collection

To obtain a wide sample of popular and unpopular websites across various categories for our crawls, we used 1.8 million unique websites from Alexa *Top Categories*[1] as our initial seed set. We sampled the top 5.8K websites from each of the 17 categories. Additionally, we randomly sampled 5.8K websites from Alexa's top 1 million. After removing duplicates, around 100K websites remained, which we use for our crawls.

We built a crawler on top of the Chrome Remote Debugging Protocol to drive the Chrome browser. The crawler works as follows: for every website w in our list, it visits the homepage. It then proceeds to extract all links L from the homepage that points to w. Our crawler randomly visits 15 links from L. If $|L| < 15$, our crawler tries to crawl $15-|L|$ additional links from the visited pages. It keeps doing so until it has crawled a total of 15 pages from w, or there are no more links to crawl. To make our crawlers look realistic, we crawled using a valid `User-Agent`, scrolled pages to the bottom, and waited ~60 seconds between subsequent page visits.

[1]https://www.alexa.com/topsites/category/Top

Table 1: High-level statistics for our crawls.

Crawl Dates	% Sites w/ Sockets	% Sockets w/ A&A Initiators	# Unique A&A Initiators	% Sockets w/ A&A Receivers	# Unique A&A Receivers
Apr 02–05, 2017	2.1	60.6	75	73.7	16
Apr 11–16, 2017	2.4	61.3	63	74.6	18
May 07–12, 2017	1.6	60.2	19	69.7	15
Oct 12–16, 2017	2.5	63.4	23	63.7	18

Overall, we performed four crawls over our sampled 100K websites. Two crawls were performed just prior to the release of Chrome 58 (which included the patch for the WRB bug) [48] between April 2–5 and April 11–16, 2017. To observe if the patch affected the usage of WebSockets by websites and A&A companies, we ran two more crawls after the release of Chrome 58. The first of these crawls was performed right after the patch between May 07–12, 2017. The second crawl was performed between October 12–16, 2017. Table 1 shows the high-level statistics for our study.

4 ANALYSIS

In this section, we analyze our dataset to understand the usage of WebSockets, the A&A companies involved, and the content being sent and received over the sockets.

4.1 Overall WebSocket Usage

We provide an overview of WebSocket usage in Table 1. We observe that only ~2% of the websites use WebSockets (column 2), with 6–12 connections on average per website that uses WebSockets.

Among the WebSockets we observe, >90% contact a third-party domain (i.e., the WebSocket was cross-origin) and 64–75% contact an A&A domain (column 5). Across all four crawls, 382 unique third-party domains and 20 A&A domains are contacted through WebSockets. Similarly, 60–63% of the WebSockets are initiated by a resource from an A&A domain (column 3). In total, we observe resources from 94 unique A&A domains initiating WebSockets.

We observe that *A&A initiators* (entities starting a WebSocket connection) and *receivers* (entities accepting a WebSocket connection) are involved in an order of magnitude more WebSocket connections than non-A&A initiators and receivers. Furthermore, we see that A&A initiators contacted only a few partners, whereas more than 47% of the A&A receivers are contacted by ≥10 parties.

Publishers. To answer how widespread WebSocket usage is among publishers, we plot Figure 3. We analyze the fraction of A&A and non-A&A WebSockets observed over publishers sorted by Alexa rank. We find that the fraction of A&A sockets is twice that of non-A&A sockets across all ranks. We also find that both types of WebSockets were most prevalent on highly-ranked domains, with a drop occurring between 10K and 20K. The fraction of A&A sockets in top 10K publishers is 4.5 times higher than that of non-A&A sockets. This demonstrates that WebSocket usage, especially A&A

Table 2: Top 15 WebSocket initiators sorted by the total number of unique receivers. A&A initiators are in bold.

	# Receivers		
Initiator	Total	A&A	Socket Count
facebook	35	11	441
espncdn	35	0	92
h-cdn	30	0	39
doubleclick	29	9	250
slither	25	0	33
google	23	11	381
youtube	18	8	129
cloudflare	15	1	873
addthis	14	8	101
hotjar	13	7	2407
googlesyndication	10	6	71
adnxs	8	3	31
googleapis	7	0	157
sharethis	6	4	20
twitter	6	5	21

Table 3: Top 15 A&A WebSocket receivers, sorted by the total number of unique initiators.

	# Initiators		
Receiver	Total	A&A	Socket Count
intercom	156	16	5531
33across	57	19	1375
zopim	44	12	19656
realtime	41	27	1548
smartsupp	26	4	670
feedjit	25	10	3013
inspectlet	25	6	820
pusher	22	8	634
hotjar	17	11	2249
disqus	17	13	4798
freshrelevance	10	2	403
lockerdome	10	8	408
velaro	4	3	62
truconversion	3	2	298
simpleheatmaps	1	0	93

Table 4: Top 15 initiator/receiver pairs communicating via WebSockets. A&A domains are in bold.

Initiator	Receiver	Socket Count
webspectator	**realtime**	1285
google	**zopim**	172
blogger	**feedjit**	158
hotjar	**intercom**	144
clickdesk	**pusher**	125
cdn77	**smartsupp**	122
acenterforrecovery	**intercom**	114
facebook	**zopim**	112
vatit	**intercom**	110
plymouthart	**intercom**	108
welchllp	**intercom**	105
biozone	**intercom**	101
getambassador	**pusher**	101
rubymonk	**intercom**	98
googleapis	sportingindex	96
A&A domain to itself		36,056

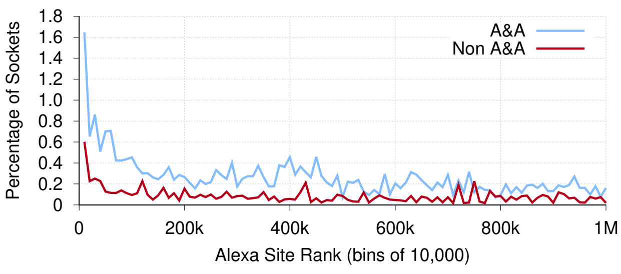


Figure 3: WebSocket usage by Alexa site rank.

sockets, is not a rarity confined to long-tail publishers; in fact, A&A sockets are widespread amongst top publishers.

Before and After. Chrome rolled out the patch for WRB in version 58 on April 19, 2017. To understand if the release of the patch affected the usage of WebSockets, we can compare the statistics in Table 1 from our crawls before and after this date. Although we see that there has been a significant drop in the number of unique A&A domains initiating WebSockets over time (column 4), the fraction of A&A-initiated sockets has essentially remained the same (column 3). In total, 56 A&A initiators disappeared between our first and last crawl, including DoubleClick, Facebook, and AddThis. It is unclear why these major advertising companies abandoned WebSockets.

With respect to A&A socket receivers, Table 1 shows that there has been essentially no change over time (column 6). As we show in § 4.2, many of the A&A receivers provide services that are dependent on WebSockets (e.g., real-time commenting and chat), thus it is not surprising that these companies have not altered their software.

4.2 Initiators and Receivers

Next, we take a deeper look into the domains that initiate and receive WebSockets. Table 2 shows the top 15 domains whose resources initiate WebSockets; A&A domains are shown in bold. We observe that scripts from some of the largest players in the online advertising ecosystem (e.g., DoubleClick, Facebook, and AppNexus) create WebSockets to multiple other A&A domains. This demonstrates that major ad exchanges have embraced WebSockets, although as noted above, some have discontinued this practice. In § 4.3, we discuss what data was being sent and received over the WebSockets by these big players.

Table 3 shows top 15 A&A domains that we observe receiving WebSocket connections. Only 2.5% of the initiators that create WebSockets to these domains are A&A domains, meaning many of the incoming connections are initiated by benign domains, or even first-party publishers. These WebSockets are particularly problematic with respect to the WRB: since the scripts that initiated the WebSockets were not blocked by AdBlock Plus, the only way to stop these connections would be to block the WebSockets themselves. To confirm this, we used the EasyList and EasyPrivacy rule lists to determine if scripts in the inclusion chains leading to A&A sockets would have been blocked. We find that only ~5% of these A&A chains would have been blocked. In contrast, ~27% of A&A chains in our overall dataset are blocked by these rule lists.[2]

In contrast to the initiators in Table 2, the A&A receivers are less well-known companies that provide a variety of services. The most recognizable company, Disqus, provides user comment boards as a service to publishers; it is also an ad network that enables publishers to monetize their comment boards by displaying targeted ads. 33across and Lockerdome are advertising platforms. Inspectlet, Hotjar, TruConversion, and SimpleHeatmaps are *session replay* services that track user interactions within websites to generate detailed analytical heatmaps of mouse movements, click, and keystrokes [17]. Zopim, Velaro, Smartsupp, and Intercom provide customer service live-chat widgets. The variety of business models offered by the receivers in Table 2 reveals an important point: WebSockets are being used to serve advertisements **and** to track users.

Table 4 shows the top 15 initiator/receiver pairs that created A&A sockets (i.e., one or both of the parties must be an A&A domain), sorted by total WebSockets. We aggregate cases where the initiator

[2]We compare the rule lists to our chains post-hoc, which may miss some requests that would have been blocked at load-time in the browser. Furthermore, these rule lists whitelist some URL patterns to avoid site breakage [16].

Table 5: Items being sent by and received by A&A domains via WebSockets and HTTP/S.

	WebSockets		HTTP/S	
Sent Item	**Count**	**%**	**Count**	**%**
User Agent	40,231	100.0	99,942,662	100.0
Cookie	28,122	69.90	22,752,063	22.77
IP	2,662	6.62	896,162	0.90
User ID	1,731	4.30	1,116,111	1.12
Device	1,453	3.61	177,101	0.18
Screen	1,443	3.59	104,794	0.10
Browser	1,368	3.40	89,614	0.09
Viewport	1,366	3.40	336,704	0.34
Scroll Position	1,366	3.40	291	0.00
Orientation	1,366	3.40	71	0.00
First Seen	1,366	3.40	8,148	0.01
Resolution	1,366	3.40	132,742	0.13
Language	722	1.79	914,628	0.92
DOM	654	1.63	8,587	0.01
Binary	396	0.98	6,267	0.01
No data	7,176	17.84	-	-
Received Item	**Count**	**%**	**Count**	**%**
HTML	18,976	47.16	11,599,601	11.61
JSON	5,152	12.81	1,633,849	1.63
JavaScript	356	0.88	27,027,458	27.04
Image	126	0.31	21,324,840	21.34
Binary	99	0.25	496,929	0.50
No data	8,580	21.33	-	-

and receiver are the same and present the total in the last row of Table 4. Unsurprisingly, the vast majority of A&A sockets fall into this category (e.g., we observe 19,064 WebSockets initiated by Zopim to themselves). The cases where the initiator and receiver are different are more interesting, in the sense that these pairs of companies chose to interface via WebSockets. These cases are also more troubling from a privacy perspective, since the WRB may have prevented blockers from halting information flows to third-parties (if the initiator's script was not blocked in the first place).

4.3 Content Analysis

In this section, we investigate the content of messages being sent and received over the WebSockets. For sent messages, we would like to know if any Personally Identifiable Information (PII) or fingerprinting-related browser state are being sent, since A&A domains can use this information to track users [2, 19, 31, 43, 51]. For received messages, we are interested in whether ad images or JavaScript (that can be used to further exfiltrate data or retrieve ads) are being downloaded.

Sent Data. Table 5 shows different items that we observe being sent and received over the A&A sockets. For comparison, we also present statistics on how frequently we observed those same items being sent/received over HTTP/S to any A&A domain. Many of the items, such as user-agents, cookies, and IP addresses, are self-explanatory. "User ID" refers to unique identifiers related to the user such as *Account ID*, *Client ID*, and *User ID* itself. "Browser" contains the fingerprinting variables used to identify *Browser Type* and *Browser Family*, whereas "Device" refers to *Device Type* and *Device Family*. "First seen" references a date field frequently seen in messages, which we believe to be the user cookie creation date. We extracted all of these variables from raw network traffic by manually building up a large library of regular expressions.

In all cases, we observe a greater percentage of private information being exfiltrated via WebSockets than over HTTP/S. This includes stateful-tracking data such as cookies, IP addresses, and unique identifiers. We also observed two particularly troubling forms of data being exfiltrated via WebSockets:

- **Fingerprinting:** ~3.4% of WebSockets exfiltrated fingerprinting data (e.g., screen size and orientation). 60 initiator/receiver pairs were involved in this practice, with 33across being the receiver in 97% of the pairs.
- **DOM Exfiltration:** In ~1.6% of WebSockets, the entire DOM was serialized and uploaded to Hotjar, LuckyOrange, or TruConversion, for the purposes of enabling session replays of user activity [17]. The DOM is potentially very privacy-sensitive, as it may reveal search queries, unsent messages, etc., within the given webpage.

We were unable to decode binary-encoded data being sent over 1% of the WebSockets. The results in Table 5 highlight that the WRB allowed trackers to circumvent blockers and implement aggressive tracking techniques.

We noted in § 4.2 that major companies like DoubleClick and Facebook stopped initiating WebSocket connections after the Chrome 58 release. We further discuss this odd coincidence in § 6. We observe that, prior to discontinuing this practice, these two companies were sending sensitive data to A&A receivers. For example, DoubleClick was sending fingerprinting data to 33across, which lists DoubleClick as one of their advertising partners [1].

Received Data. Next, we examine the information received over A&A sockets. Of the 78.7% WebSockets that did receive any data, WebSockets downloaded a greater percentage of HTML and JSON, as compared to JavaScript and images which were downloaded more often over HTTP/S.

We did not observe any ad images being sent directly over WebSockets (we checked for binary and base64 encoded media files). However, we did find that Lockerdome was sending URLs to ad images in their WebSocket responses, along with meta-data such as image captions, heights, and widths. These images were hosted on `cdn1.lockerdome.com`, which was not blacklisted in EasyList, meaning that the WRB was effectively allowing Lockerdome to circumvent ad blockers. Figure 4 shows three examples of these ads, which are emblematic of the low-quality "clickbait" that is served by unscrupulous ad networks and Content Recommendation Networks [9]. Furthermore, these are the same types of ads that were flagged by users in the WRB bug reports [27, 50]. This demonstrates that there are ad networks who were willing to exploit the WRB to serve ads, and that unsurprisingly, these shady ad networks cater to shady advertisers.

5 RELATED WORK

The Online Ad Ecosystem. There are a plethora of empirical studies that have measured the online advertising ecosystem. Barford et al. [7] looked at the major ad networks on the web by mapping the online *adscape*, whereas Rodriguez et al. measured the

Figure 4: Example of ads received over WebSockets. *Left*: "Odd Trick To Fix Sagging Skin". *Center*: "Study Reveals What Just A Single Diet Soda Does To You". *Right*: "Win an iPad Air 2 from Addicting Games!"

ad ecosystem on mobile devices [55]. Gill et al. [24] used browsing traces to study the economy of online advertising and discovered that most of the revenue is skewed towards a few big companies. Acar et al. [2] conducted crawls over the Alexa Top-3K to find user identifiers being shared across domains. Similarly, Cahn et al. [12] observed that <1% of the trackers are present on 75% of Alexa Top-10K websites. Falahrastegar et al. [20] looked at online trackers across geographic regions.

Other empirical studies have focused on the individual implications of targeted advertising. Guha et al. [25] developed a controlled method for measuring online ads on the web based on trained *personas*. Carrascosa et al. [13] used these methods to prove that advertisers use sensitive attributes about users when targeting ads. Bashir et al. used retargeted ads to determine information flows between ad exchanges [8], and demonstrate how close collaboration among advertisers affect user privacy [10]. Olejnik et al. [46] noticed winning bid prices being leaked during Real Time Bidding (RTB) auctions and used this information to investigate the relative value of different users.

Researchers have also studied bad practices in the advertising ecosystem. Zarras et al. [57] studied ad networks running malicious ad campaigns, whereas Bashir et al. [9] found some advertisers not following industry guidelines and serving poor quality ads.

Tracking Mechanisms. Krishnamurthy et al. were one of the first to bring attention to the pervasiveness of trackers and their privacy implications [32]. Since then, several studies have documented the evolution of online tracking on the web [12, 18, 33, 34, 36].

Advertisers have upgraded their tracking techniques over time. Some of the techniques they employ include persistent cookies [30], local state in browser plugins [6, 53], browsing history through extensions [54], and fingerprinting methods [2, 18, 19, 31, 40, 43, 45, 51]. Some studies have investigated information sharing through cookie matching [2, 8, 21, 46].

Anti-tracking. To avoid pervasive tracking, users are increasingly adopting tools that block trackers and ads [37, 49]. Papaodyssefs et al. [47] proposed the use of private cookies to mitigate tracking, while Nikiforakis et al. added entropy to the browser to combat fingerprinting [42]. Merzdovnik et al. and Iqbal et al. performed large scale measurements of blocking extensions and techniques to determine which are most effective [28, 39].

Snyder et al. [52] performed a browser feature usage survey and showed that ad and tracking blocking extensions do not block all standards equally, with WebSockets being blocked 65% of the times. Franken et al. [22] reported that blocking extensions could sometimes be bypassed using WebSockets. They found that the extention developers made the mistake of using "`http://*, https://*`" filters instead of "`ws://*, wss://*`" for the `onBeforeRequest` event, which prevents the interception of WebSocket connections.

To the best of our knowledge, there is no prior work which provides an in-depth analysis of WebSocket usage across the web.

6 DISCUSSION

The Good. Overall, our measurements demonstrate that the WRB was not leveraged to circumvent blockers by the vast majority of A&A companies. Although we find that ~68% of WebSockets on the open web are initiated or received by A&A domains (see § 4.1), most of these companies have a legitimate reason to use WebSockets. For example, Disqus, Zopim, and Intercom all offer real-time services that are ideal use-cases for WebSockets (see § 4.2).

The Strange. A troubling finding of our study is that major ad and tracking platforms, like Google, Facebook, AddThis, and AppNexus adopted WebSockets (see Tables 2 and 4). This is extremely concerning, since these companies dominate the online display ad ecosystem and are essentially omnipresent on the web. Furthermore, we observe these companies sharing sensitive fingerprinting data over the WebSockets. Yet strangely, we do not observe these major ad platforms initiating WebSocket connections after the release of Chrome 58 (when the WRB was patched, see § 4.1). The observational nature of our study prevents us of from drawing causal conclusions about this finding, and indeed, it may be coincidental.

The Bad. Previous studies of online tracking have repeatedly identified "innovators" attempting to use bleeding-edge techniques to gain an advantage against privacy-conscious users. Examples include the use of persistent cookies and various kinds of fingerprinting [2, 6, 18, 19, 30, 31, 40, 43, 45, 51, 53, 54].

We identify five companies that appear to have been using the WRB to circumvent blocking extensions: 33across was harvesting large amounts of browser state that could be used for fingerprinting; Lockerdome was downloading URLs to ads (see § 4.3 and Figure 4); and Hotjar, LuckyOrange, and TruConversion were downloading the entire DOM from webpages.

Conclusion. Our work demonstrates the lengths A&A companies are willing to go to counter blocking techniques, and highlights the importance of measurement studies to keep up with the current practices of A&A companies. With respect to the WRB, users clamored for a patch after observing ads slipping through blockers [27, 50], but our results demonstrate that invisible tracking was an equally important and disturbing implication of the WRB.

Our results add a longitudinal perspective to controlled studies that have identified other, novel techniques to bypass privacy-preserving measures in browsers [22]. This body of work emphasizes the importance of patching browser vulnerabilities (in a reasonable amount of time) that A&A companies can leverage to their advantage. More broadly, this suggests that browser vendors need to prioritize bug reports not just through the lens of usability and security, but also privacy.

ACKNOWLEDGMENTS

We thank our shepherd, John Byers, and the anonymous reviewers for their helpful comments. This research was supported in part by NSF grants CNS-1563320 and IIS-1553088. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

[1] 33across. [n. d.]. ATTENTION PLATFORM FOR ADVERTISERS. 33across. `https://33across.com/attention-platform/`.

[2] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. 2014. The Web Never Forgets: Persistent Tracking Mechanisms in the Wild. In *Proc. of CCS*.

[3] Adblock Plus 2014. WebSocket connections can't be blocked. AdBlock Plus Issue Tracker. `https://issues.adblockplus.org/ticket/1727`.

[4] AdChoices 2017. Put the YourAdChoices Icon to Work for You. Digital Advertising Alliance. `http://youradchoices.com/learn`.

[5] Sajjad Arshad, Amin Kharraz, and William Robertson. 2016. Include Me Out: In-Browser Detection of Malicious Third-Party Content Inclusions. In *Proc. of Intl. Conf. on Financial Cryptography*.

[6] Mika Ayenson, Dietrich James Wambach, Ashkan Soltani, Nathan Good, and Chris Jay Hoofnagle. 2011. Flash cookies and privacy II: Now with HTML5 and ETag respawning. *Available at SSRN 1898390* (2011).

[7] Paul Barford, Igor Canadi, Darja Krushevskaja, Qiang Ma, and S. Muthukrishnan. 2014. Adscape: Harvesting and Analyzing Online Display Ads. In *Proc. of WWW*.

[8] Muhammad Ahmad Bashir, Sajjad Arshad, , William Robertson, and Christo Wilson. 2016. Tracing Information Flows Between Ad Exchanges Using Retargeted Ads. In *Proc. of USENIX Security Symposium*.

[9] Muhammad Ahmad Bashir, Sajjad Arshad, and Christo Wilson. 2016. "Recommended For You": A First Look at Content Recommendation Networks. In *Proc. of IMC*.

[10] Muhammad Ahmad Bashir and Christo Wilson. 2018. Diffusion of User Tracking Data in the Online Advertising Ecosystem. In *Proc. of PETS*.

[11] BugReplay. 2016. Pornhub Bypasses Ad Blockers With WebSockets. medium.com. `https://medium.com/thebugreport/pornhub-bypasses-ad-blockers-with-websockets-cedab35a8323`.

[12] Aaron Cahn, Scott Alfeld, Paul Barford, and S. Muthukrishnan. 2016. An Empirical Study of Web Cookies. In *Proc. of WWW*.

[13] Juan Miguel Carrascosa, Jakub Mikians, Ruben Cuevas, Vijay Erramilli, and Nikolaos Laoutaris. 2015. I Always Feel Like Somebody's Watching Me: Measuring Online Behavioural Advertising. In *Proc. of ACM CoNEXT*.

[14] Chrome Debugging Protocol [n. d.]. Chrome DevTools Protocol Viewer. GitHub. `https://developer.chrome.com/devtools/docs/debugger-protocol`.

[15] Stacy Cowley and Julianne Pepitone. 2012. Google to pay record $22.5 million fine for Safari privacy evasion. CNNMoney. `http://money.cnn.com/2012/08/09/technology/google-safari-settle/index.html`.

[16] EasyList [n. d.]. EasyList Policy. The EasyList authors.. `https://easylist.to/pages/policy.html`.

[17] Steven Englehardt. 2017. No boundaries: Exfiltration of personal data by session-replay scripts. Freedom to Tinker Blog. `https://freedom-to-tinker.com/2017/11/15/no-boundaries-exfiltration-of-personal-data-by-session-replay-scripts`.

[18] Steven Englehardt and Arvind Narayanan. 2016. Online tracking: A 1-million-site measurement and analysis. In *Proc. of CCS*.

[19] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W. Felten. 2015. Cookies That Give You Away: The Surveillance Implications of Web Tracking. In *Proc. of WWW*.

[20] Marjan Falahrastegar, Hamed Haddadi, Steve Uhlig, and Richard Mortier. 2014. The Rise of Panopticons: Examining Region-Specific Third-Party Web Tracking. In *Proc of. Traffic Monitoring and Analysis*.

[21] Marjan Falahrastegar, Hamed Haddadi, Steve Uhlig, and Richard Mortier. 2016. Tracking Personal Identifiers Across the Web. In *Proc. of PAM*.

[22] Gertjan Franken, Tom Van Goethem, and Wouter Joosen. 2018. Who Left Open the Cookie Jar? A Comprehensive Evaluation of Third-Party Cookie Policies. In *Proc. of USENIX Security Symposium*.

[23] Masahiro Fujimoto. 2018. Introduction to the DOM. Mozilla. `https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction`.

[24] Phillipa Gill, Vijay Erramilli, Augustin Chaintreau, Balachander Krishnamurthy, Konstantina Papagiannaki, and Pablo Rodriguez. 2013. Follow the Money: Understanding Economics of Online Aggregation and Advertising. In *Proc. of IMC*.

[25] Saikat Guha, Bin Cheng, and Paul Francis. 2010. Challenges in Measuring Online Advertising Systems. In *Proc. of IMC*.

[26] Raymond Hill. [n. d.]. A companion extension to uBlock Origin. GitHub. `https://github.com/gorhill/uBO-Extra`.

[27] Raymond Hill. 2016. ws-gateway websocket circumvention ? #1936. GitHub. `https://github.com/gorhill/uBlock/issues/1936`.

[28] Umar Iqbal, Zubair Shafiq, and Zhiyun Qian. 2017. The Ad Wars: Retrospective Measurement and Analysis of Anti-Adblock Filter Lists. In *Proc. of IMC*.

[29] Issue 129353 2012. chrome.webRequest.onBeforeRequest doesn't intercept WebSocket requests. Chromium Bugs. `https://bugs.chromium.org/p/chromium/issues/detail?id=129353`.

[30] Samy Kamkar. 2010. Evercookie - virtually irrevocable persistent cookies. `http://samy.pl/evercookie/`.

[31] T. Kohno, A. Broido, and K. Claffy. 2005. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing* 2, 2 (2005), 93–108.

[32] Balachander Krishnamurthy, Delfina Malandrino, and Craig E. Wills. 2007. Measuring Privacy Loss and the Impact of Privacy Protection in Web Browsing. In *Proc. of SOUPS*.

[33] Balachander Krishnamurthy, Konstantin Naryshkin, and Craig Wills. 2009. Privacy Diffusion on the Web: A Longitudinal Perspective. In *Proc. of WWW*.

[34] Balachander Krishnamurthy and Craig Wills. 2011. Privacy leakage vs. Protection measures: the growing disconnect. In *Proc. of W2SP*.

[35] Tobias Lauinger, Abdelberi Chaabane, Sajjad Arshad, William Robertson, Christo Wilson, and Engin Kirda. 2017. Thou Shalt Not Depend on Me: Analysing the Use of Outdated JavaScript Libraries on the Web. In *Proc of NDSS*.

[36] Adam Lerner, Anna Kornfeld Simpson, Tadayoshi Kohno, and Franziska Roesner. 2016. Internet Jones and the Raiders of the Lost Trackers: An Archaeological Study of Web Tracking from 1996 to 2016. In *Proc. of USENIX Security Symposium*.

[37] Matthew Malloy, Mark McNamara, Aaron Cahn, and Paul Barford. 2016. Ad Blockers: Global Prevalence and Impact. In *Proc. of IMC*.

[38] Mapx. 2016. ws-gateway websocket circumvention? GitHub. `https://github.com/gorhill/uBlock/issues/1936`.

[39] Georg Merzdovnik, Markus Huber, Damjan Buhov, Nick Nikiforakis, Sebastian Neuner, Martin Schmiedecker, and Edgar R. Weippl. 2017. Block Me If You Can: A Large-Scale Study of Tracker-Blocking Tools. In *Proc. of Euro S&P*.

[40] Keaton Mowery and Hovav Shacham. 2012. Pixel perfect: Fingerprinting canvas in HTML5. In *Proc. of W2SP*.

[41] Muhammad Haris Mughees, Zhiyun Qian, and Zubair Shafiq. 2017. Detecting Anti Adblockers in the Wild. *PoPETs* 2017, 3 (2017), 130.

[42] Nick Nikiforakis, Wouter Joosen, and Benjamin Livshits. 2015. PriVaricator: Deceiving Fingerprinters with Little White Lies. In *Proc. of WWW*.

[43] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. 2013. Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting. In *Proc. of IEEE Symposium on Security and Privacy*.

[44] Rishab Nithyanand, Sheharbano Khattak, Mobin Javed, Narseo Vallina-Rodriguez, Marjan Falahrastegar, Julia E. Powles, Emiliano De Cristofaro, Hamed Haddadi, and Steven J. Murdoch. 2016. Adblocking and Counter Blocking: A Slice of the Arms Race. In *Proc. of FOCI*.

[45] Lukasz Olejnik, Claude Castelluccia, and Artur Janc. 2012. Why Johnny Can't Browse in Peace: On the Uniqueness of Web Browsing History Patterns. In *Proc. of HotPETs*.

[46] Lukasz Olejnik, Tran Minh-Dung, and Claude Castelluccia. 2014. Selling off Privacy at Auction. In *Proc of NDSS*.

[47] Fotios Papaodyssefs, Costas Iordanou, Jeremy Blackburn, Nikolaos Laoutaris, and Konstantina Papagiannaki. 2015. Web Identity Translator: Behavioral Advertising and Identity Privacy with WIT. In *Proc. of HotNets*.

[48] pkalinnikov. [n. d.]. Issue 2449913002: Support WebSocket in WebRequest API. (Closed). Chromium Code Reviews. `https://codereview.chromium.org/2449913002/`.

[49] Enric Pujol, Oliver Hohlfeld, and Anja Feldmann. 2015. Annoyed Users: Ads and Ad-Block Usage in the Wild. In *Proc. of IMC*.

[50] Rhana Joy 2016. TechnoBuffalo.com. EasyList Forum. `https://forums.lanik.us/viewtopic.php?p=110902`.

[51] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. 2012. Detecting and Defending Against Third-party Tracking on the Web. In *Proc. of NSDI*.

[52] Peter Snyder, Lara Ansari, Cynthia Taylor, and Chris Kanich. 2016. Browser Feature Usage on the Modern Web. In *Proc. of IMC*.

[53] Ashkan Soltani, Shannon Canty, Quentin Mayo, Lauren Thomas, and Chris Jay Hoofnagle. 2010. Flash Cookies and Privacy.. In *AAAI Spring Symposium: Intelligent Information Privacy Management*.

[54] Oleksii Starov and Nick Nikiforakis. 2017. Extended Tracking Powers: Measuring the Privacy Diffusion Enabled by Browser Extensions. In *Proc. of WWW*.

[55] Narseo Vallina-Rodriguez, Jay Shah, Alessandro Finamore, Yan Grunenberger, Konstantina Papagiannaki, Hamed Haddadi, and Jon Crowcroft. 2012. Breaking for Commercials: Characterizing Mobile Advertising. In *Proc. of IMC*.

[56] Steven J. Vaughan-Nichols. 2017. Chrome is the most popular web browser of all. ZDNet. `http://www.zdnet.com/article/chrome-is-the-most-popular-web-browser-of-all/`.

[57] Apostolis Zarras, Alexandros Kapravelos, Gianluca Stringhini, Thorsten Holz, Christopher Kruegel, and Giovanni Vigna. 2014. The Dark Alleys of Madison Avenue: Understanding Malicious Advertisements. In *Proc. of IMC*.