Ad-Blocking and Counter Blocking: A Slice of the Arms Race

Nithyanand, R; Khattak, S; Javed, M; Vallina-Rodriguez, N; Falahrastegar, M; Powles, JE; Cristofaro, ED; Haddadi, H; Murdoch, SJ

Adblocking and Counter-Blocking: A Slice of the Arms Race

Rishab Nithyanand[1], Sheharbano Khattak[2], Mobin Javed[3],
Narseo Vallina-Rodriguez[4], Marjan Falahrastegar[5], Julia E. Powles[2],
Emiliano De Cristofaro[6], Hamed Haddadi[5], Steven J. Murdoch[6]

[1] Stony Brook University [2] University of Cambridge [3] University of California - Berkeley
[4] International Computer Science Institute [5] Queen Mary University of London [6] University College London

Abstract

Adblocking tools continue to rise in popularity, altering the dynamics of advertising revenue streams. In response, a number of publishers have ramped up efforts to develop and deploy tools for detecting and/or counter-blocking adblockers (tools we refer to as *anti-adblockers*), effectively escalating the online advertising arms race. In this paper, we leverage a novel approach for identifying third-party services shared across multiple websites to present a first characterization of anti-adblocking across the Alexa Top-5K websites. We map websites that perform anti-adblocking as well as the entities that provide anti-adblocking scripts. We study the modus operandi of these scripts and their impact on popular adblockers. We find that at least 6.7% of websites in the Alexa Top-5K use anti-adblocking scripts, acquired from 12 distinct entities – some of which have a direct interest in nourishing the online advertising industry.

1 Introduction

Today's web ecosystem is largely driven by online advertising. However, recent years have seen a large number of users turn to adblocking and tracker-blocking tools[1] [13] for the purposes of improving their web-browsing experience, maintaining privacy, and more recently to protect themselves against malvertising [16]. With a recent study estimating the number of active adblock users to be 198M and revenue losses due to adblockers at $22B [14], the threat posed by adblockers to the online advertising revenue model has moved from mildly concerning to existential. In response, publishers have started to actively detect users of adblockers, and subsequently block them or otherwise coerce them to disable the adblocker– in the rest of the paper, we refer to this practice as *anti-adblocking*. Most recently, this practice gained wide attention with the endorsement of the Internet Advertising Bureau (IAB) when, in March 2016, it released a primer on how to deal with users of adblockers, as well as semi open-source script[2] for detecting the use of adblockers [10]. The tension between key stakeholders in this ecosystem – publishers, users, and a plethora of intermediate beneficiaries (whether categorised as ad-tech, adblocking, anti-adblocking or micropayment) – forms part of what has been dubbed as the *adblocking arms race* [17].

Motivation. While incidents of anti-adblocking [2, 5, 15,17], and the legality of such practices [3,12,19], have received increasing attention, our current understanding is limited to online forums [2] and user-generated reports [5]. As a result, we lack quantifiable insights into key questions such as: how prevalent nowadays are such practices on the Web? Are certain categories of websites more likely to employ anti-adblockers? Who are

[1] While adblocking differs from tracker-blocking, to ease presentation, we refer to tools that provide any of these properties as adblockers.

[2] The script was only made available to members of the IAB.

the main suppliers of anti-adblocking services? What mechanisms do these employ to detect the presence of adblockers? Is it possible for adblockers to counter-block anti-adblockers? What are common responses after positive detection of adblockers and their impact on end-user? In this work, we address these questions by presenting the first characterization of anti-adblocking.

Roadmap. We start with characterizing anti-adblocking on the Web by identifying anti-adblocking scripts across Alexa Top-5K sites. To this end, we develop a scalable novel technique to identify popular third-party services that are shared across multiple websites, and utilize it to flag anti-adblocking scripts. We then map out the entities that serve anti-adblocking scripts and the websites that use these scripts. We find that at least 6.7% of Alexa Top-5K websites conduct some form of anti-adblocking by downloading 14 scripts from 12 unique domains most of which belong to ad services, while one specifically offers anti-adblocking services. Most of the anti-adblocking websites represent popular categories such as news, blogs, and entertainment. We manually visit sample websites from the anti-adblockers and find that the arms race has already entered the next round: at least one of three popular browser extensions (AdBlock Plus, Ghostery, Privacy Badger) can counter-block half of the anti-adblocking scripts. We conclude with a discussion of the anti-adblocking arms race in terms of ethics and legality, also enumerating existing proposals that aim to achieve a sustainable and unintrusive online advertising model.

2 Methodology

This section presents our method for identifying third-party services that are shared between multiple websites. We describe the technique in the context of identifying shared anti-adblocking JavaScripts (JS). The premise of our approach is that by discovering *similar* objects (in our case, JavaScripts) that are loaded by multiple websites, we can infer the presence of a common third-party JS, its functionality and its source.

Crawler overview. We rely on *Crawler Incantatus*, a Selenium-based interactive crawler used by the ICLab [8], to generate the set of JavaScripts to analyze. We load each website in our dataset with four browser modes – vanilla Firefox, Firefox with AbBlock Plus, Firefox with Ghostery, and Firefox with Privacy Badger. For each page load, we capture screenshots, HTML source code, and responses to all requests generated by the browser. We extract all the text between `<script>` and `</script>` from the HTML and tag them as *embedded* JS. Similarly, we detect all JS objects in the collected responses and tag them as *downloaded* JS. In total, the Top-5K Alexa websites generate over 200K individual JS files.

Identifying JS objects with common sources. We formulate our problem of finding groups of similar JS as a maximal clique finding problem [4]. We consider each JS file loaded by a website to be a node in a graph. If two nodes are within some margin of similarity of each other (we define our similarity metric below), we say there is an edge between them. We extract classes of JS that have a common source by identifying all maximal cliques in this graph. By intentionally focusing on finding similar JS (rather than identical JS) we allow for the grouping of objects that differ only slightly because they contain website-specific identifiers and properties.

Choice of similarity metric and threshold. In order to add an edge between two nodes in the graph (i.e., to indicate that two JS files are similar), we need to define a metric for similarity, and a suitable threshold for this metric. To measure the *similarity* of two JS files, we use Term Frequency–Inverse Document Frequency (TF-IDF) to generate a vector of *keyword weights* for each JS file after filtering out JS reserved words, such as `function` and `var`. We then use the cosine similarity metric to measure the similarity of the two keyword weight vectors. Similar approaches using both TF-IDF and cosine similarity have been used by the information retrieval community for topic identification and similarity checking of source-code [9, 20]. We note that this method is particularly well suited to our task compared to other string matching approaches because it is:

- *White-space insensitive:* Many websites perform script minification using different libraries, yielding different indentation and white-spacing practices. Our approach is unaffected by these complications.
- *Position insensitive:* In scripts that have several functionalities (e.g., tracking and ad-block detection), the position of each specific function is irrele-

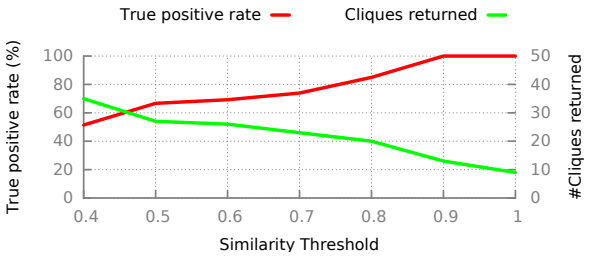

Figure 1: Effect of the similarity threshold parameter on the True Positive Rate (TPR) and the number of maximal cliques.

vant to the similarity score.

- *Reasonably resistant to noise:* Small changes (e.g., website specific identifiers) have little impact on the final similarity score.

In order to determine a similarity score *threshold*, we perform a series of clique extraction experiments on a small dataset of 4.4K JS files extracted from the Alexa Top-100 websites. We find, via manual inspection, that at a threshold of 0.80, 17/20 cliques returned by our program contain scripts with identical functionality and sources, i.e., achieving True Positive Rate (TPR) of 0.85. In Figure 1, we illustrate the change in TPR along with the number of cliques returned as the threshold increases. Although thresholds above 0.90 yield TPR=1.0, the number of cliques returned drops significantly, which will result in lower True Negative Rates (TNR). Therefore, following a conservative stance, we use a threshold of 0.80 for the remainder of our experiments.

Improving scalability. Our approach involves computing the cosine similarity between each pair of keyword weight vectors, thus requiring $O(n^2)$ vector multiplications for n JS files. Given the large number of JS files used by websites (e.g., the Alexa Top-5K sites contained over 200K JavaScripts), this may not scale with large datasets. Therefore, we use a set of heuristically developed filters to eliminate comparisons between scripts that are unlikely to ever be part of the same clique:

- *Word-count filter:* We avoid comparing scripts with significant word-count difference. Specifically, if a pair of scripts has a word-count ratio higher than 1.50, we assume that they are unlikely to be a part of the same clique and set their similarity to 0.
- *Embedded vs. downloaded script filter:* JavaScript is either embedded in the source HTML for page-specific functionality, or downloaded separately from external sources to provide site-wide functionality. We do not consider them as the same type of identity thus we set their similarity to 0.
- *Source filter:* If two JavaScripts are downloaded from the exact same URL, we mark them as identical.
- *JS domain filter:* JavaScript can communicate with external sources indicated by embedded URLs. We assume that for any pair of scripts, if one communicates with external sources and the other does not, their functionality is different and set their similarity score to 0.

Source and functionality identification. Once maximal cliques of similar scripts are identified, the content and meta-data of each script in a clique is used to generate and log: (*i*) the FQDN (Fully Qualified Domain Name) of the script's source, (*ii*) FQDNs of external resources utilized by the script, and (*iii*) keywords associated with the script. In Section 3, we use these three features, in addition to content of the script, to classify cliques by functionality.

Method limitations. We acknowledge that our method has a number of limitations: Notably, our similarity metric will fail to identify obfuscated JS code. Furthermore, given that we do not compare downloaded with embedded JS code, we may fail to identify small cliques in which a reduced number of sites integrate an anti-adblocking JS in a different way than is normal. Additionally, our method may fail to identify similarities between composed JS– i.e., scripts that consist of multiple individual files downloaded as a single object. As a result, our method only provides a lower-bound approximation of the usage of anti-adblocking across websites. We plan on addressing these limitations in future work.

3 Dataset and Results

We apply our clique detection methodology to the JS objects fetched by our crawler. We restrict our analysis to cliques of size greater than 5 – i.e., JavaScripts shared

	Downloaded	Embedded
Cliques	1373	509
Websites	3619	2070
Total websites: 4017		
	Trackers	**Anti-Adblockers**
Cliques	456	22
Websites	2741	335

Table 1: The number of total cliques and those related to tracking and anti-adblocking, along with the number of websites that incorporate these scripts (computed over 200K downloaded and embedded scripts).

19.5%	General News	2.5%	Pornography
9.3%	Blogs/Wiki	2.5%	Forum/Bulletin Boards
8.5%	Entertainment	2.2%	Technical/Business Forums
4.3%	Internet Services	2.2%	Potential Illegal Software
3.7%	Sports	2.0%	Online Shopping
3.7%	Games	1.7%	Portal Sites
3.2%	Travel	1.7%	Humor/Comics
3.2%	Education/Reference	1.2%	Social Networking
2.7%	Business	1.2%	Provocative Attire
2.5%	Software/Hardware	1.2%	Marketing/Merchandising

Table 2: Distribution of anti-adblocking websites by category according to McAfee's URL categorization.

by more than 5 sites in our dataset – as we are interested in identifying scripts that are shared across multiple websites. We acknowledge that this approach might fail to flag anti-adblocking scripts utilized by individual or a small number of websites, and those used by a few websites in the Alexa Top-5K but popular among websites ranked above 5K. As shown in Table 1, we find 1,373 cliques that are shared among 3,619 websites in the downloaded files, with an average of 232 websites per clique (σ=365.6) and the largest clique having 1,320 websites (which we find, via manual inspection, is a JS related to user interface). Among the embedded scripts, 509 cliques are shared by 2,070 websites (μ=41.2 σ=48.9 max=261). We manually analyze all the 1,882 cliques (corresponding to 4,017 unique websites) identified for both downloaded and embedded scripts, and tag them as *trackers* (if they upload information such as IP addresses and cookies to tracking companies), *anti-adblockers* (if they check for the presence of adblockers), or *others*. We note that manual analysis of JS is a tedious process and does not scale to a large number of scripts. In future work, we plan on looking into ways to automate JS tagging.

We uncover 22 cliques used for anti-adblocking employed by 335 websites – about 6.7% of Alexa Top-5K websites. We observe that Alexa Top-1K have 60 anti-adblocking websites, and the number increases by about 70 websites for every additional 1K considered, reaching 335 anti-adblocking websites in Top-5K. While studying anti-adblockers, we also identify 456 tracking cliques employed by about 54% of Alexa Top-5K, validating previous studies on the pervasiveness of tracking over the Web [6].

Anti-adblocking by website categories. In Table 2, we report the categories of the 335 anti-adblocking websites, using McAfee's URL categorization service [11]. We find that anti-adblocking is common among a diverse mix of publishers, and prevalent among publishers of "General News" (19.5%), "Blogs/Wiki" (9.3%), and "Entertainment" (8.5%) categories, which represent more than one third of all websites. Note that these categories are also among the most popular ones across all Top-5K Alexa domains, although to a lesser extent – respectively, 9.4%, 6.29%, and 5.4%. Whereas, other popular categories among Top-5K domains (e.g., "Internet services", "Online Shopping", "Business", which account for 20% of the Top-5K) are much less prevalent in anti-adblocking websites.

Website response to detection of adblockers. In order to assess how anti-adblocking websites behave once they identify adblockers, we look at all the screenshots taken by our crawler, respectively, *without* any adblocking extension and *with* AdBlock Plus (which we assume is more likely to be detected due to its popularity [13]).

We note cases where there is an explicit (i.e., warning to disable adblocker) or a discrete (i.e., blank page via AdBlock Plus, but normal appearance without) response to adblocking. For these websites, we also view screenshots when accessed with Ghostery, Privacy Badger, and NoScript enabled.

We find only 6 explicit and no discrete responses to adblocking. Of the explicit responses, 3 are displayed by porn websites hosted by the same company – `MindGeek` – and employ the same anti-adblocking

script downloaded from DoublePimp. The warning is displayed for both AdBlock Plus and Ghostery. The remaining 3 also employ the same script, but display different messages (only for AdBlock Plus) with the same general theme, i.e., nudging the user to disable the adblocker and/or support the website via subscription or donation.

Some websites display adblocker warning to users after they engage in some form of activity, such as clicking on links or scrolling. To capture such responses, we repeat the above exercise for screenshots taken after mimicking user activity – specifically, clicking on a random link on the page, scrolling down to the bottom of the newly loaded page, waiting three seconds, then scrolling back up to the top of the page, waiting 5 seconds. While the modified methodology validates our previous observations, we do not discover any new responses.

In the attempt of automating the analysis of websites' response to anti-adblocking, we have also tried to use image comparison tools, such as perceptual hashing. However, this generates a high number of false positives due to dynamic content on many sites as well as false negatives since anti-adblocking warnings and messages generate a relatively small visual difference.

How anti-adblockers work. Next, we manually inspect the 22 anti-adblocking scripts (14 downloaded and 8 embedded) aiming to understand how anti-adblocking scripts detect adblockers. We note that of these only the 14 downloaded scripts are actually useful as the 8 embedded scripts simply redirect to the downloaded scripts. We find that anti-adblockers operate on a simple premise: if an expected advertisement-related element on the publisher's website is missing when the page loads, the script concludes that the user has an adblocker installed.

Specifically, the anti-adblocker detects adblockers by injecting a dummy advertisement container element (e.g., DIV), and then comparing the values of properties representing dimensions (height and width) and/or visual status (display) of the container element with the expected values when properly loaded. To track whether the user has turned off the adblocker after being prompted to do so, the anti-adblocker periodically runs the ad-block check and stores the last recorded status in the user's browser using a cookie or local storage.

Domain	Description	#Sites	ABP	Gh	PB
pagefair.com	Anti-adblocking	20	✓	✗	✓
googleadservices.com	Ads	61	✗	✗	✗
googlesyndication.com	Ads	13	✗	✗	✗
taboola.com	Ads	36	✗	✓	✓
outbrain.com	Ads	10	✗	✓	✓
ensighten.com	Ads	6	✗	✓	✗
hotjar.com	Analytics	9	✗	✗	✗
doublepimp.com	Pornography	8	✗	✓	✗
tacdn.com	Travel	8	✗	✗	✗
cloudflare.com	CDN	50	✗	✗	✓
cloudfront.net	CDN	6	✗	✗	✗
ytimg.com	Content/Ads	108	✗	✗	✗

Table 3: Domains from which anti-adblocking scripts are downloaded and #websites employing them. The table's right side reports whether AdBlock Plus, Ghostery, and Privacy Badger counter-block anti-adblocking scripts from these domains.

Anti-adblocker suppliers. We analyze the source code of the 14 anti-adblocking scripts and the domains from which these are downloaded aiming to infer the suppliers of these scripts. The remaining 8 embedded scripts redirect to anti-adblocking scripts served by Cloudflare and Taboola. Our analysis is summarized in Table 3. We also include a description of these domains – based on the information available on their official websites, Google search, and Mcafee URL categorization service [11] – as well as the number of websites in our dataset that employ the anti-adblocker.

At the top we find Pagefair, a company specialized in anti-adblocking services, followed by a number of domains related to Google, Taboola, Outbrain and Ensighten. Overall the anti-adblockers downloaded from these 5 domains are employed by 48% of all the 315 websites employing anti-adblockers. We note that these domains are direct beneficiaries of anti-adblocking as these inherently thrive on the prevalence of online advertisements. Though not directly related to online advertisement, the ability to detect adblockers is a useful capability for the analytics company HotJar.

We also find two cases where the anti-adblocking script is shared by entities in the same domain or business: TripAdvisor (*tacdn.com*) distributes the script to its 8 websites with different country code top-level

domains. Adult websites, all of which are hosted by `MindGeek`, turn to `DoublePimp` for anti-adblocking. Two anti-adblocking scripts are pulled from popular Content Delivery Networks (CDNs), but we could not determine their original supplier. Finally, `ytimg` (a content server associated with YouTube) serves a script that has the ability to detect if ads were properly loaded, however, it is not clear how it uses this information.

Adblocker response to being blocked. There is anecdotal evidence that the adblocking arms race has entered the next level: some adblockers can detect anti-adblockers and counter-block them [18]. To test for this behaviour, we visit a sample website for each anti-adblocking script via AdBlock Plus, Ghostery and Privacy Badger over Chrome web browser. We repeat the experiment three times and monitor all HTTP requests generated when loading the website using Chrome's *Developer Tools*. We infer that adblocker can counter-block if the request to fetch anti-adblocking script fails to be initiated. As reported in Table 3, half of the 12 anti-adblocking suppliers are blocked by at least one adblocker. Ghostery and Privacy Badger detect 4 anti-adblockers each, while AdBlock Plus detects only 1. Anti-adblocking scripts served by `Taboola` and `Outbrain` are blocked by both Ghostery and Privacy Badger, `PageFair` scripts by both AdBlock Plus and Privacy Badger, while `Doublepimp`, `Ensighten` and `Cloudflare` scripts by at most one of the three adblockers. We note that the anti-adblocking suppliers that are never detected are related to content distribution, Google ad services, analytics, or site-wide scripts.

4 Discussion

The adblocking arms race involves a plethora of players: between publishers and consumers, a jostling array of intermediaries compete to deliver ads, mostly supported by business models that involve taking a cut of the resultant advertising revenue. At the heart of this rich ecosystem lie important questions regarding the legality and ethics of adblocking and anti-adblocking.

The legality of adblocking is potentially contestable under laws about anti-competitive business conduct and copyright infringement. To date, only Germany has tested these arguments in court, with adblockers winning most [3], but not all of the cases [12]. On the other hand, anti-adblocking in the EU might in turn breach Article 5(3) of the Privacy and Electronic Communications Directive 2002/58/EC, as it involves interrogating an end-user's terminal equipment without consent [19].

Many consider adblocking to be an ethical choice for consumers and publishers to consider from both an individual and societal perspective. In reality, however, both sides have resorted to radical measures to achieve their goals. The Web has empowered publishers and advertisers to track, profile and target users in a way that is unprecedented in the physical realm [6]. In addition, publishers are inadvertantly and increasingly serving up malicious ads (malvertisements) [16]. This has resulted in the rise of adblocking, which in turn has led publishers to employ anti-adblocking. The core issue is to get the balance right between ads and information. Publishers turn to anti-adblocking to force consumers to reconsider the default blocking of ads for earnest ad-supported publishers. Defaults are very difficult to shift at scale. Nevertheless, those publishers will fail if they do not redress in a fundamental way the reasons that brought consumers to adblockers in the first place. There exist proposals to provide a compromise, such as privacy-friendly advertising [7] as well as mechanisms to give users more control over ads and trackers they are exposed to [1, 21]. Our work extends these efforts by providing quantified insights into anti-adblocking, to inform policy that improves upon the current blocking/counter-blocking deadlock.

5 Conclusions

In this paper, we presented a measurement-based analysis providing a first look at anti-adblocking in the online advertising arms race. We found that at least 6.7% of Alexa Top-5K websites, mostly in popular categories like news, blogs, and entertainment, engage in some form of anti-adblocking. The arms race has already entered the next level, as at least one of three popular browser extensions – AdBlock Plus, Ghostery, Privacy Badger – can evade half of the anti-adblocking scripts in our dataset. In future work, we plan to extend our measurements beyond the Alexa Top-5K websites, and experiment with crowdsourced and/or automated mech-

anisms to tag JavaScript by functionality and to assess publisher response to detection of adblockers.

References

[1] J. P. Achara, J. Parra-Arnau, and C. Castelluccia. MyTrackingChoices: Pacifying the Ad-Block War by Enforcing User Privacy Preferences. In *WEIS*, 2016.

[2] Adblock Plus. Filters for Adblock Plus forum. https://adblockplus.org/forum/viewforum.php?f=2&sid=83e35818f92df8cf921623f2ff27ce70.

[3] Adblock Plus. Five and oh look, another lawsuit upholds users' rights online. https://adblockplus.org/blog/five-and-oh-look-another-lawsuit-upholds-users-rights-online.

[4] C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, 16(9), 1973.

[5] Campaign Against The Illegal Detection/Circumvention Of Adblocking Tools. Alexander Hanff. https://adblocking.think-privacy.com.

[6] M. Falahrastegar, H. Haddadi, S. Uhlig, and R. Mortier. Tracking Personal Identifiers Across the Web. In *PAM*, 2016.

[7] S. Guha, B. Cheng, and P. Francis. Privad: Practical privacy in online advertising. In *NDSI*, 2011.

[8] ICLab. http://www.iclab.org.

[9] A. Kuhn, S. Ducasse, and T. Gírba. Semantic clustering: Identifying topics in source code. *Information and Software Technology*, 49(3), 2007.

[10] I. T. Lab. IAB Tech Lab Publisher Ad Blocking Primer. Technical report, 2016.

[11] McAfee. http://www.trustedsource.org.

[12] MEEDIA. Doppelte Attacke: Springers zwei Fronten-Strategie im Kampf gegen Ad-Blocker. http://meedia.de/2015/12/15/doppelte-attacke-springers-zwei-fronten-strategie-im-kampf-gegen-ad-blocker/.

[13] Mozilla. Firefox: Most Popular Extensions. https://addons.mozilla.org/en-us/firefox/extensions/?sort=users.

[14] PageFair. The 2015 Ad Blocking Report. https://blog.pagefair.com/2015/ad-blocking-report/.

[15] Schneier on Security. The Ads vs. Ad Blockers Arms Race. https://www.schneier.com/blog/archives/2016/02/the_ads_vs_ad_b.html, 2016.

[16] The Guardian. Major sites including New York Times and BBC hit by 'ransomware' malvertising . https://www.theguardian.com/technology/2016/mar/16/major-sites-new-york-times-bbc-ransomware-malvertising, 2016.

[17] The New York Times. The Ad Blocking Wars. http://nyti.ms/1Qs20YB, 2016.

[18] The Next Web. This adblocker-blocker helps you get around sites that ban you for hiding ads. http://thenextweb.com/apps/2016/02/11/around-and-around-we-go/#gref, 2016.

[19] The Register. Ad-blocker blocking websites face legal peril at hands of privacy bods. http://www.theregister.co.uk/2016/04/23/anti_ad_blockers_face_legal_challenges/.

[20] T. Yamamoto, M. Matsushita, T. Kamiya, and K. Inoue. Measuring similarity of large software systems based on source code correspondence. In *Product Focused Software Process Improvement*. Springer, 2005.

[21] Z. Yu, S. Macbeth, K. Modi, and J. M. Pujol. Tracking the Trackers. In *WWW*, 2016.