Investigating Advertisers' Domain-changing Behaviors and Their Impacts on Ad-blocker Filter Lists

Su-Chin Lin*
Kai-Hsiang Chou*
National Taiwan University
Taiwan
{r07922067,b07705022}@ntu.edu.tw

Yen Chen
National Taiwan University
Taiwan
b06902047@ntu.edu.tw

Hsu-Chun Hsiao
National Taiwan University
Academia Sinica
Taiwan
hchsiao@csie.ntu.edu.tw

Darion Cassel
Carnegie Mellon University
United States
darioncassel@cmu.edu

Lujo Bauer
Carnegie Mellon University
United States
lbauer@cmu.edu

Limin Jia
Carnegie Mellon University
United States
liminjia@cmu.edu

ABSTRACT

Ad blockers heavily rely on filter lists to block ad domains, which can serve advertisements and trackers. However, recent research has reported that some advertisers keep registering *replica ad domains* (RAD domains)—new domains that serve the same purpose as the original ones—which tend to slip through ad-blocker filter lists. Although this phenomenon might negatively affect ad blockers' effectiveness, no study to date has thoroughly investigated its prevalence and the issues caused by RAD domains. In this work, we proposed methods to discover RAD domains and categorized their change patterns. From a crawl of 50,000 websites, we identified 1,748 unique RAD domains, 1,096 of which survived for an average of 410.5 days before they were blocked; the rest have not been blocked as of February 2021. Notably, we found that non-blocked RAD domains could extend the timespan of ad or tracker distribution by more than two years. Our analysis further revealed a taxonomy of four techniques used to create RAD domains, including two less-studied ones. Additionally, we discovered that the RAD domains affected 10.2% of the websites we crawled, and 23.7% of the RAD domains exhibiting privacy-intrusive behaviors, undermining ad blockers' privacy protection.

CCS CONCEPTS

• **Security and privacy → Browser security**; **Web application security**.

KEYWORDS

domain-changing behavior, replica ad domain, filter list, ad blocking

*Both authors contributed equally to this research.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512218

ACM Reference Format:
Su-Chin Lin, Kai-Hsiang Chou, Yen Chen, Hsu-Chun Hsiao, Darion Cassel, Lujo Bauer, and Limin Jia. 2022. Investigating Advertisers' Domain-changing Behaviors and Their Impacts on Ad-blocker Filter Lists. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3485447.3512218

1 INTRODUCTION

Ad blockers are software programs designed to block advertisements and trackers from websites. Using ad blockers can improve user experience during web browsing, such as reducing unwanted information and accelerate page loading time, and protect users' privacy, such as hiding device fingerprints or blocking tracking scripts. Currently, many ad blockers use filter lists [7], which are lists of rules that cover known ad domains[1], to identify advertisements and trackers. For example, EasyList and EasyPrivacy [25] are two filter lists used by many popular ad blockers such as Adblock Plus, uBlock Origin, and Brave. [26] Compared with advanced techniques that dynamically detect advertisements [13, 37, 38, 42] or tracking scripts [16, 33, 41], filter-list-based ad blockers are widely adopted and well maintained, and can alleviate browsers from the burden of runtime analysis.

However, the static rules in filter lists certainly cannot cover all newly registered ad domains. Recent research and actual observations have discovered that some advertisers keep registering new domains to serve the same advertisements or trackers over time. In this paper, we call these kinds of domains *replica ad domains (RAD domains)*. For example, Yalvi, Propellerads, and PopAds are known to create new RAD domains using Domain Generation Algorithms (DGA) [3, 11]. Intentionally or not, RAD domains produced by such domain-changing behaviors often slip through the rules of ad blockers' filter lists. Although this phenomenon might negatively affect the effectiveness of ad blockers, to the best of our knowledge, no prior study has systematically investigated its influence on the widely used ad-blocker filter lists. Several previous studies [11, 56] were limited to analyzing those that have already been identified

[1]For ease of presentation, because most ad blockers block both advertisements and trackers, we use *ad domains* to refer to domains serving advertisements, tracking scripts, or both.

by filter lists instead of those newly changed domains. A few researchers noted the existence of advertisers' domain-changing behaviors [11, 15, 36, 56], but they did not present a thorough analysis.

To understand the impacts of advertisers' domain-changing behaviors on ad-blocker filter lists, we present our research that answers the following research questions:

- RQ1: What are the common patterns of domain-changing behaviors performed by the advertisers?
- RQ2: What are the RAD domains' prevalence and survival time before being added to filter lists?
- RQ3: Among the RAD domains we discovered, how many of them exhibit privacy-intrusive behaviors?

To address those limitations of existing approaches, this paper explores methods to search for RAD domains in the wild, rather than within known ad domains. It proceeds from the observation that, an advertiser exhibiting the domain-changing behavior may control two domains $\mathcal{A}$ and $\mathcal{B}$ that serve similar advertising or tracking content, and change from a blocked domain $\mathcal{A}$ to another domain $\mathcal{B}$ that has not been blocked. In such cases, we refer to $\mathcal{B}$ as a *RAD domain* for $\mathcal{A}$, and $\mathcal{A}$ is a *related ad domain* for $\mathcal{B}$. Because they share the same owner and have similar functionality, a RAD domain may leave linkable traces to its related ad domain. More precisely, we determine whether two domains have the same owner via DNS records and TLS certificates, and determine whether they have similar functionality via URL paths and their content.

Among the 252,601 unique domains that we encountered while crawling 50,000 websites, we identified 1,748 RAD domains; 1,096 of them were known ad domains as of February 2021, and we manually validated the remaining. Our analysis revealed several common domain-changing patterns, including *moving to first-party subdomains* (17.4%), *using revolving domains* (12.7%), *changing subdomains* (35.9%), and *using CDN domains* (9.6%). We further discussed whether ad blockers could effectively identify and block each of these patterns. For example, by reviewing EasyList forum discussions, we found cases suggesting that filter-list operators favor specific rules for blocking individual subdomains over "wildcard" rules blocking entire domains to avoid false positives. The EasyList policy also explicitly states that this service is lenient about first-party ad domains [27]. Thus, we speculate that advertisers take advantage of such attitudes and policies when circumventing ad blockers. Moreover, because using first-party subdomains to proxy ads essentially abuses users' trust in the first-party websites, this may severely affect users' security and privacy if the ad domain is malicious or compromised.

Our analysis showed that these RAD domains affected 10.2% of the websites we crawled. It took 424.2 days on average for a RAD domains to be found and blocked since their first appearances, and non-blocked RAD domains extended the timespan of ad or tracker distribution by more than two years. Additionally, about 23.7% of the RAD domains exhibited privacy-intrusive behaviors, according to the Tracker Radar [23]. These findings suggest that RAD domains can severely harm users' privacy because users can still be tracked even with ad blockers enabled.

We also discussed possible reasons behind the use of RAD domains. Besides ad-blocker circumvention, we presented cases whose primary purposes seem to be for client isolation, localization, or infrastructure upgrades.

This work makes the following contributions:

- We proposed methods for discovering RAD domains in the wild and confirmed that the ad-blocker filter lists did not block a substantial number of them as of February 2021.
- We presented a taxonomy of common domain-changing patterns, which could help the community develop countermeasures and improve filter rules.
- We revealed that the RAD domains affected 10.2% of the websites we crawled, and 23.7% of them were privacy-intrusive.
- We discussed possible reasons behind the use of RAD domains.
- Our dataset and analysis scripts will be published to help future researchers and the filter-list community.[2]

2 METHODOLOGY

This section presents our methods to investigate advertisers' domain-changing practices, including how we discovered domain-changing events (§2.1), what data and how they were collected (§2.2), and our analysis methods (§2.3). The limitations of our methodology are discussed in Appendix A.

2.1 Domain-changing events

At a high level, to discover domain-changing events, we searched for and analyzed *RAD domains* of known ad domains. A domain $\mathcal{B}$ is a RAD domain of an ad domain $\mathcal{A}$ if they have the **same owner** and **similar functionalities**, and we say $\mathcal{A}$ is a *related ad domain* of $\mathcal{B}$. We utilized the DNS records and TLS certificates to discover the same-owner relationships, and URL paths and served files to discover similar-functionality relationships. Additionally, as all RAD domains by definition should be ad domains, we manually validated and removed non-ad domains to reduce false positives. We also excluded domains that were blocked at the time of their first encounter because they would not have been seen by ad-blocker users and have no impact on ad-blocker users' browsing experience and privacy. That is, all RAD domains in our analysis are ad domains that once successfully bypassed the ad-blockers.

2.1.1 Identification of same-owner domains. An intuitive approach to inferring a domain's owner is to retrieve registrant information from the WHOIS database. However, we did not use WHOIS to recover the domain owner because the WHOIS database contains only TLD+1, and many domains hide their registrant information for privacy reasons, especially following the implementation of the General Data Protection Regulation [6, 35]. Therefore, instead of trying to discover the domain owner, we looked for other evidence of whether two domains were owned by the same entity. For this purpose, a pair of domains was considered to have the same owner if linked by either DNS records or TLS certificates. As Appendix B.1 confirms, both methods complement each other, as each led to some discoveries of RAD domains that the other did not.

DNS records. DNS is a hierarchical system for associating a domain name to an IP address. Only the domain owner, as registered with a registrar, can modify its own DNS records. In most cases, a domain owner configures an *A* or *AAAA* record resolved to the IP address of its own machine, or a *CNAME* record mapping

[2]The full list of RAD domains and the analysis scripts can be access at https://github.com/csienslab/RAD-domain-analysis/.

the domain to an alias name. If two domains are hosted on the same machine, their DNS records will be resolved to the same IP eventually. Since an IP is typically controlled by a single owner who is the only entity able to modify its DNS records, we can infer whether two domains have the same owner from their DNS records. (Exceptions, such as CDN and cloud hosting, are discussed in Appendix A.1.) It is possible to link two domains to each other via the three above-mentioned types of DNS records, i.e., *CNAME*, *A*, and *AAAA*. A *CNAME* record points to an alias domain, and *A* and *AAAA* records represent IPv4 and IPv6 addresses, respectively. We therefore deemed domains $\mathcal{A}$ and $\mathcal{B}$ to have the same owner if (1) both had an identical *CNAME*, *A*, or *AAAA* record, or (2) one domain's *CNAME* record pointed to the other.

TLS certificates. A TLS certificate is a signed document that binds a public key to one or more domains. To obtain a valid certificate, a domain owner has to prove the ownership of all the domains listed in the *Common Name* (CN) and *Subject Alternative Name* (SAN) fields in the certificate. Thus, if two domains appear on the same certificate, they likely have the same owner. Our approach of using TLS certificates was inspired by Cassel et al. [15].

2.1.2 Identification of similar-functionality domains. To find domains with similar functionality, we used (1) *URL paths* to link domains having similar endpoints or (1) *served files* to link domains serving a similar set of files. If either one was satisfied, we inferred that the two domains have similar functionality.

URL paths. To link domains having similar endpoints, we quantified the closeness between two request URL paths $P_{\mathcal{B}}$ and $P_{\mathcal{A}}$ using a *SIM* function, defined as: $SIM(P_{\mathcal{A}}, P_{\mathcal{B}}) = IPF(LCP(P_{\mathcal{A}}, P_{\mathcal{B}}))$. The function $LCP(P_{\mathcal{A}}, P_{\mathcal{B}})$ returns the longest common path for $P_{\mathcal{A}}$ and $P_{\mathcal{B}}$. For example, $LCP(/foo/bar, /foo/baz)$ is evaluated to $/foo$. Similar to *Inverse Document Frequency*, the $IPF(p)$ function, which stands for *Inverse Path Frequency*, computes the logarithmically scaled inverse fraction of unique domains that have the URL path. Our approach to doing so was similar to Kargaran et al.'s [42] use of Inverse Document Frequency metric [40, 60], but we assign a higher weight to a less-visited path [62]. For instance, common paths like `/js` and `/static` (visited by 23,830 and 8,524 domains, respectively) carry little information compared with `/site_media` (visited by just 28 domains) because they are more common seen among all collected domains.

We considered that $\mathcal{A}$ and $\mathcal{B}$ have similar functionalities if they satisfy the following inequality: $\exists\, P_{\mathcal{A}}, P_{\mathcal{B}}, SIM(P_{\mathcal{A}}, P_{\mathcal{B}}) \geq \alpha$, where $P_{\mathcal{A}}$ and $P_{\mathcal{B}}$ are URL paths of $\mathcal{A}$ and $\mathcal{B}$, respectively. Additionally, $P_{\mathcal{A}}$ and $P_{\mathcal{B}}$ should not contain words typically used for directory brute-forcing [61], thereby excluding paths that are common but seldom visited, such as `/wp-admin`. The choice of the threshold α is discussed in Appendix D.1.

Served files. We inferred that two domains have similar functionality if they served more than a threshold of identical resources. We set the threshold to avoid falsely linking two domains that serve an identical file (e.g., a well-known JavaScript library) by coincidence, in contrast to Snyder et al. [56]'s method, which links two domains sharing at least one identical file. Specifically, we compared non-script files via their hashes. For script files, inspired by Chen et al.'s method [16], we performed code analysis by generating the abstract syntax tree (AST) using Esprima [2] and extracting the node types. Then, if two files shared the same AST, or one AST was the subtree of another, we consider them the same script. By this method, our analysis is resilient to trivial changes, such as renaming variable or combining multiple scripts into one file [16].

We compared the files served by $\mathcal{A}$ and $\mathcal{B}$ and filtered out 1KB or smaller files to prevent trivially identical contents. In the end, we considered $\mathcal{A}$ and $\mathcal{B}$ to have similar functionality if the Jaccard similarity between their sets of served files was higher than 0.7. These parameters were selected based on our small-scale experiment that showed that most tracking pixels and trivial content, such as error messages, are smaller than 1KB, while scripts and visible images are often larger. We empirically set a Jaccard similarity threshold so that domains serving widely used scripts are not linked. The further justification is in Appendix D.2.

2.1.3 Manual validation. After identifying same-owner and similar-functionality domains, we obtained a set of potential RAD domains, but some might be false positives. One common cause of false positives is domain parking: When an ad domain expires and is parked, it will have the same DNS records and served files as all other parked domains, and thus be linked together. As all RAD domains by definition should be ad domains, we removed non-ad domains to reduce false positives. We manually validated the potential RAD domains that were not on our ad-domain dataset (i.e., were not blocked by the end of February 2021), following EasyList's policy. In most cases, the verification was trivial. For example, identical ad-related script files or multimedia files served, some keywords (e.g., the advertisers' name) appeared in the scripts, or the scripts contained suspicious fingerprinting APIs. In some cases, we made our best effort to verify by visiting the first-party websites (or the corresponding snapshots) and examining the requests and content [24].

2.2 Data collection

We created a domain dataset containing 252,601 unique domains and an ad-domain dataset of 61,824 known ad domains. For each domain in these two datasets, we collected their DNS records, TLS certificates, served files, and URL paths. The latter two were obtained at the same time as we crawled the domain datasets, and thus this subsection only describes the collection of domain dataset, ad-domain dataset, DNS records, and TLS certificates.

Domain dataset. Our domain dataset is effectively a "pool" of domains within which we searched for RAD domains. To increase the odds that this dataset would contain RAD domains of known ad domains, we crawled historical snapshots of websites and collected all the unique domains encountered during crawling.

We created a list of 50,000 websites comprising (1) Tranco's top 20,000 and (2) another 30,000 randomly sampled from among those ranked 20,001st to 1,000,000th by Tranco [43]. This approach allowed us to include lesser-known websites in our dataset. We then crawled the historical snapshots of each of these 50,000 websites using Wayback Machine [34]. We stored one snapshot per website per month for the period from March 2015 to February 2021, inclusive.

Ad-domain dataset. We collected 61,824 unique known ad domains from four sources of filter lists used by two popular adblockers, AdGuard AdBlocker and uBlock Origin. These four sources of

filter lists are EasyList [28], EasyPrivacy [28], AdGuard [9], and Peter-Lowe's Blocklist [49].

DNS records. We collected the DNS records in May and June 2021 because we were not aware of any affordable and reliable services that would have allowed us to query such historical records for more than 100,000 domains. We collected 648,128 DNS records for 254,689 domains by querying Google Public DNS [30].

TLS certificates. We retrieved the certificates used by each domain in both our datasets between March 2015 and February 2021 from `crt.sh` [53]. For each certificate, we extracted the domains listed in the CN and SAN fields for later analysis. This process resulted in us collecting 217,979 certificates in all.

2.3 Analysis methods

2.3.1 Timestamp reconstruction. To assess filter lists' effectiveness against RAD domains, we defined two time-based metrics, *survival time* and *additional survival time*. A domain's *survival time* is defined as the time passes between its appearing time and blocked time. The survival time indicates how fast the filter list maintainers find and block the RAD domain. A domain's *additional survival time* is defined as the time elapsed between $\mathcal{B}$'s appearing time or $\mathcal{A}$'s blocked time (whichever is later) and $\mathcal{B}$'s blocked time. The additional survival time indicates how much the advertisers extend the timespan of ads distributions by utilizing RAD domains.

To compute these metrics, we need to estimate each domain's *appearing time* and *blocked time*. We estimated each domain's appearing time by its earliest snapshot available on Wayback Machine. However, that time-point does not necessarily equate to the moment a site was brought online; i.e., Wayback Machine might or might not snapshot a site as soon as it is available, resulting in the estimated time of appearance later than the actual one. In other words, the actual (additional) survival time could be longer, implying that our findings may be more conservative than the reality.

We estimated each one's blocked time based on the four lists' historical versions from March 2015 to February 2021, as obtained from their official GitHub repositories and websites. To simplify our analysis, in case where a domain was added, removed, and added again, we used the first time it was added as its blocked time.

2.3.2 Ad-domain clusters. If an advertiser changes its ad domains multiple times, we would like to be able to group them into an *ad-domain cluster* for further analysis, as the size of the ad-domain clusters can provide insights into the characteristics and usages of RAD domains. For example, an advertiser that keeps changing its domains will likely result in a larger cluster. To cluster ad domains (including all known ad domains and the new ones we found in this study), we constructed a graph in which nodes are ad domains, and a directed edge from node x to y means that y is a RAD domain of x, i.e., changing from x to y. An ad-domain cluster is a weakly connected component on the graph.

3 RESULTS

Our Wayback Machine crawler yielded 2,165,106 snapshots of 50,000 websites, with 252,601 unique domains visited, of which 49,490 were known ad domains. We identified 1,748 RAD domains from all domains visited using our proposed methods. Among them, 1,096 were blocked as of February 2021, and 652 of them were not blocked but labeled as ad domains after manual validation.

3.1 Common domain-changing patterns

Examining the relationships among RAD domains, their related ad domains, the ad-domain clusters they belong to, and their first parties, we categorized four common patterns among the 1,748 RAD domains, where the last two are relatively less-studied previously, as summarized below.

- **Moving to first-party subdomains**: 305 RAD domains were hosted on their first-party subdomains. Among them, 175 were linked to third-party ad domains by CNAME, sometimes referred to as CNAME cloaking [19];
- **Using revolving domains**: 222 RAD domains were generated using DGA by advertisers known to constantly create new domains for evasion, known as revolving domains [1];
- **Changing subdomains**: 627 RAD domains shared parent domains with their related ad domains;
- **Using CDN domains**: 167 RAD domains were hosted on known CDN domains.

We present each pattern in turn and discuss the challenges of blocking them using ad-blocker filter lists. We provide general and pattern-specific recommendations in Section 4.2.

3.1.1 Moving to first-party subdomains. Among the 1,748 RAD domains, we found 309 located under their first-party websites.

For instance, `dynamic-js.compass.com` was a RAD domain of a known ad domain `tracking.keywee.co` , and only appeared on its first-party domain `compass.com`. While the first party is a real estate platform, this subdomain is under control by Keywee Inc., a marketing company. This is strong evidence that the first-party domain delegates its subdomains to proxy ads.

To further understand this pattern, we surveyed several advertiser and tracker websites [17, 48, 50, 52, 54], and found detailed tutorials explaining how to configure a first-party custom proxy to delegate ads, thus bypassing ad-blocker filter lists. For example, Paradot [48] (a tracker service) and Plausible [50] (a website analytics provider) explained how to delegate ads using first-party subdomains. To check whether any websites had followed Paradot's advice, we searched within the DNS records we had collected and found 35 first-party subdomains containing a CNAME record `go.pardot.com`. Similarly, four websites followed Plausible's advice and moved their domains to first-party subdomains.

We speculate that some advertisers exploit EasyList's leniency regarding first-party ad domains. According to EasyList's policy [24, 27], "the subscription's policy is slightly more lenient with first-party tracking, specifying that items should be blocked only if they "collect a significant amount of personal data." Thus, a first-party ad domain may be blocked only when it performs fingerprinting [27] and collects large amounts of user information, such as IP address, user agent, screen resolution, time zone, and language.

A major concern of using first-party subdomains is that, it blurs the trust boundary between the first- and third-party entities, essentially abusing users' trust in the first-party website. Another concern is that if the third party is malicious or compromised, it may severely affect users' security and privacy, given that it operates in the context of the first-party website.

Table 1: RAD domains in Clickadu's ad-domain cluster.

RAD domain	Appearing date	Blocked date	Survival time (days)
sghehllds.com	2019-06-21	2019-07-25	33
drjgjngf.com	2019-06-22	2019-07-15	22
qumagee.com	2019-07-16	2019-10-04	79
xineday.com	2019-08-18	2019-09-05	17
mrzikj.com	2019-08-26	2019-10-09	43
tibacta.com	2019-09-01	2019-09-07	5

CNAME cloaking. CNAME cloaking [8, 10, 20, 51, 56] is a technique that create a first-party subdomain and assigns a DNS *CNAME* record pointing to the blocked ad domain. Prior work [20, 51] has observed that advertisers leverage CNAME cloaking to evade ad blockers. Among the 305 RAD domains hosted on first-party websites, we observed 175 (57.4%) utilizing this technique, and 106 of which were not blocked as of February 2021, confirming that CNAME cloaking poses challenges to ad-blocking via filter-lists.

It is difficult for ad blockers to detect CNAME cloaking because most browsers limit ad-blocker plugins from accessing resolved DNS records. In other words, these plugins only know the requested domain and have insufficient information to detect CNAME cloaking. Among mainstream desktop browsers (i.e., Chrome, Edge, Firefox, Opera, and Safari), only Firefox and Safari have the ability to detect CNAME cloaking. Firefox 60+ (released in January 2018) supports DNS API [45] allowing plugins to retrieve resolved DNS records. Starting from version 14 (released in September 2020), Safari's built-in Intelligent Tracking Prevention (ITP) can detect CNAME cloaking, and restrict cookies from the first-party subdomains using CNAME cloaking as if they are third-party resources [39]. However, Firefox's and Safari's small market share (in 2020, 7.3% for Firefox and 3.7% for Safari [47]) means that the majority of users are vulnerable to CNAME cloaking as an evasion technique.

3.1.2 Using revolving domains. Some advertisers are known to serve ads by constantly creating DGA-generated domains. The Easylist community called these *revolving domains* [1]. Yalvi [3] is one of the earliest advertisers using this technique, then Propellerads and PopAds also use similar methods [11].

By manual inspecting the ad domain clusters whose domains were all DGA-like, we found 222 revolving domains from 15 advertisers, affecting 294 first parties. The advertisers with the most revolving domains were LuckyAds (36 domains), Clickadu (35), and AdSpyGlass (27). Moreover, there was a cluster with 51 revolving domains of which we could not confirm the owner. Some revolving domains from Clickadu are presented in Table 1.

The countermeasure the EasyList community took was to block all third-party requests on the websites that were known to be using Yalvi [3], which later expanded to all ad networks deploying revolving domains. This method, however, may break the functionality of the websites, and it relies on the comprehensive list of the websites deploying revolving domains. As far as we know, this policy is still in effect. Another route the EasyList community took is to automatically discover new revolving domains by monitoring the websites known to be using them [5]. This method has two defects. First, the default expired time of EasyList is four days, which means that even if the new revolving domain is discovered immediately after its deployment, it remains accessible to users for four days in the worst case. Also, by delivering different ad domains to different first parties, the advertisers can serve ads without being noticed by the automation tools. Our data suggest that there were pools of revolving domains being served concurrently to different first parties. For example, in August 2020, there were 20 revolving domains served by AdSpyGlass at the same time, ten of which appeared on distinct first-parties. Although this was merely a circumstantial proof of the evasion attempt, the strategy is technically possible.

To shed light on the efficiency of EasyList policy, we analyzed the additional survival time of the revolving domains. We reconstructed the appearance sequence of revolving domains by sorting them by their time of the first appearance, and considered the adjacent two to be pairs of the related ad domain and the RAD domain. The median of the additional survival time was 34 days, and the average was 80 days, which was longer than expected. We suspect that it might be due to the selection bias of our method, as we only considered the ad domains that bypassed the filter lists. To validate our claim, we found that 214 revolving domains were already blocked when we first encountered them, indicating that the filter-list maintainers reacted fast to nearly half of the revolving domains we encountered. Still, our findings show that the filter lists may sometimes miss revolving domains for a long time.

Another finding is that all revolving domains in our dataset were discovered via DNS records. For example, all revolving domains from LuckyAds, such as `qakdki.com` and `inpiza.com`, shared common CNAME records (either `luckylb.com` or `luckyads.tech`). Similarly, all revolving domains from another ad-domain cluster, such as `adeclc.com` and `basetts.com`, were resolved to `teser.net` via CNAME records. This finding indicates that blocking revolving domains via DNS records is promising.

3.1.3 Changing subdomains. We found 627 RAD domains (226 eTLD+1[3]) that had the same parent domains as their related ad domains, meaning that the advertisers simply changed the subdomain. For instance, `ggdata1.cnr.cn` was a RAD domain of a known ad domain `adsame1.cnr.cn`. Both were under the parent domain, `cnr.cn`. However, the ad domain was blocked by the rule `||adsame1.cnr.cn^` on July 7, 2017, and soon later, the RAD domain appeared on October 11, 2017. It was not blocked until a more general rule `||cnr.cn/s?z=`[4] was added on August 10, 2018.

Conceivably, filter-list maintainers favor specific rules blocking individual subdomains over wildcard rules blocking entire domains, to avoid false positives. Unfortunately, such an approach allows advertisers to bypass filter rules easily. An examination of the discussion threads on the EasyList forum turned up several cases supporting the above speculation. One example is `socialblade.com`. One of its subdomains, `analytics.socialblade.com`, was added to EasyPrivacy on May 24, 2019. Another subdomain, `analytics2.socialblade.com`, was added on July 3, 2019. SocialBlade started using another subdomain `cupid.socialblade.com` on December 30, 2020, and it was blocked on February 1, 2021. However, the parent domain `socialblade.com` was not on the list as of February 2021. Another example is `*.optimizely.com`. EasyPrivacy blocked this domain but included multiple exception rules (such as `cdn.optimizely.com/js/$domain=zdnet.com`) to

[3] An *effective top-level domain* (eTLD) is a domain under which users can directly register names, defined by the Public Suffix List [46]. eTLD+1 domains are more fine-grained than TLD+1 domains, in which TLD stands for top-level domains.

[4] The rule blocks requests whose paths starting with `s?z=` and the domain is `cnr.cn` or its subdomains.

remove false positives. This demonstrates that naively blocking a wildcard domain might introduce a large number of false positives. Moreover, though blocking/allowing specific path substrings can reduce false positives, advertisers can circumvent them by changing the paths, which is easier than changing domains.

3.1.4 Using CDN domains. Among the 1,761 RAD domains, five were hosted on Azure CDN (`*.azureedge.net`) and 160 on Amazon CloudFront CDN (`*.cloudfront.net`). Because CDN domains are easy to create and can serve benign content, it is reasonable for filter-list maintainers to be hesitant about blocking CDN-based ad domains. For example, EasyList and EasyPrivacy contain 327 unique rules for blocking CDN subdomains under `*.cloudfront.net`, each of which is prefixed with a different hash value, such as `dls7rxd829s2x.cloudfront.net`.

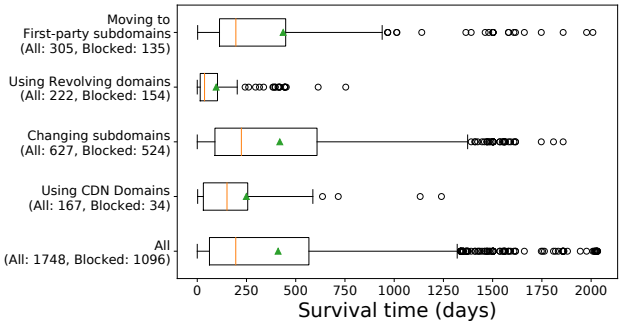

Figure 1: The box plot of the survival time of blocked domains exhibiting common patterns. The triangle marker indicates mean value.

3.2 Impact of RAD domains

In total, among all the 50,000 websites we crawled, 5,121 (10.24%) have sent requests to at least one RAD domain. Since RAD domains appeared on a non-negligible number of websites, they could substantially harm user privacy if they exhibited privacy-intrusive behaviors and survived for a long enough time. Thus, to assess the impact of RAD domains on user privacy, this subsection analyzes survival time (how long they have survived before being blocked) and additional survival time (how long the timespan of ads distribution is extended).

3.2.1 Survival time. Among all 1,096 blocked RAD domains, the average survival time was 410.5 days, and the median was 195.5 days. Only 334 RAD domains (30.5%) were blocked within 90 days from their first appearances. This may indicate that the filter-list maintainers struggled to keep up with the fast deployment of RAD domains.

Figure 1 and Table 3 also list the survival times of each common domain-changing pattern. The average survival time of RAD domains that exhibited at least one pattern was 336.4 days, and the median was 152.5 days. The survival times of *moving to first-party subdomains* and *changing subdomains* were similar to that of all RAD domains. However, only 26 of the *moving to first-party subdomains* cases (8.5%) were found and blocked within 90 days, much lower than that of all RAD domains (30.5%), indicating that the filter-list maintainers struggled to discover them in time. The

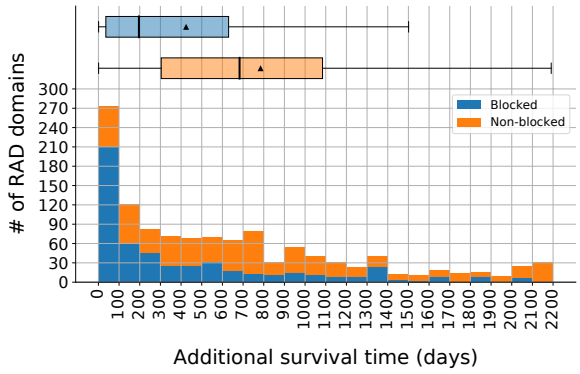

Figure 2: The histogram and box plot of additional survival time for RAD domains. The non-blocked RAD domains were assumed to be blocked by the end of February 2021.

revolving domains had the shortest survival time (95.92 days on average), and nearly half of them survived less than 90 days, reflecting the effectiveness of the automatic detection tools mentioned in Section 3.1.2. The RAD domains hosted on CDN (i.e., the *using CDN domains* pattern) had the lowest percentage of being blocked, but their average survival time was shorter than that of all RAD domains. This result shows that it is generally challenging to identify RAD domains hosted on CDN, but some easy cases may be caught quickly.

3.2.2 Additional survival time. We used *additional survival time* (defined in §2.3) to measure how much the advertisers can extend the timespan of advertisement distributions by utilizing the RAD domains. Figure 2 shows the additional survival time of blocked (in blue) and non-blocked (in orange) RAD domains as of February 2021. Note that, for a non-blocked domain, we present a *lower bound* of its additional survival time—that is, assuming that it was blocked by the end of February 2021 (i.e., the end of our Wayback Machine and block-list data collection).

For non-blocked RAD domains, their average additional survival time (based on the lower-bound estimation) was 784.7 days, suggesting the existence of long-standing RAD domains. On the other hand, for blocked RAD domains, the average additional survival time was 424.2 days, and the median was 196 days. This significant discrepancy between blocked and non-blocked RAD domains may be suggesting that the automatic detection deployed by filter-list maintainers can quickly detect and block on the monitored ad networks [11]; however, those not monitored can survive for a very long time. Moreover, as we presented in Section 3.1.2, advertisers can keep changing their resource-serving domains, such that the overall survival time can increase significantly even though each domain was shortlived.

3.3 Purpose of trackers on RAD domains

Advertisers can use ad domains to host trackers for several purposes, such as serving ads, marketing attributions, or analyzing user behaviors. We analyzed the purposes of trackers on RAD domains to understand how they were used, their impact on users, and whether they showed different distribution than the general ad domains, i.e., all the known ad domains we encountered in our crawls.

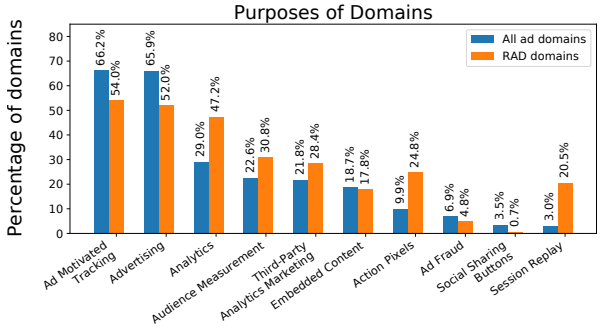

Figure 3: The percentage of ad domains and RAD domains, by purpose. The populations are ad domains (5,133) and RAD domains (420) with identifiable purposes, respectively. See Table 2 in the appendix for the explanation of each purpose.

We used the dataset from DuckDuckGo's Tracker Radar [23] to infer the purposes of trackers on RAD domains. Tracker Radar regularly crawls the Internet and extracts domains exhibiting privacy-intrusive behaviors, such as extensive uses of browser APIs known for fingerprinting. Thus, our analysis presented in this section is limited to domains with privacy-intrusive behaviors; others, such as ad domains serving only static image files, are excluded.

Table 2 in the appendix details each purpose included in our analysis. Note that one RAD domain may have several purposes; for example, most domains in *Ad Motivated Tracking* also appeared in *Advertising*. Since the Tracker Radar dataset contained only a subset RAD domains we crawled, we expanded this dataset by heuristically inferring a RAD domain's purpose via its related ad domain, and vice versa. We additionally labeled 300 domains by doing so.

We confirmed that 415 RAD domains (23.7% of all RAD domains) were serving privacy-intrusive trackers, and these RAD domains affected 1,758 first-party domains. Figure 3 shows the distribution of purposes. 54.0% of the trackers on RAD domains were related to advertisement (*Ad Motivated Tracking*), performing privacy-intrusive tasks such as targeting users and demographic collection, affecting 2.8% of the first-party websites we crawled. In addition, more than 20% of trackers on RAD domains, nearly seven times of the general ad domains, were performing session replay, which collects much more data than typical analytic scripts. This confirms that many RAD domains were serving privacy-harming contents and exhibiting privacy-intrusive behaviors. We also discuss the prevalence of browser fingerprinting in Appendix C.

4 DISCUSSION

4.1 Possible reasons of using RAD domains

A common presumption of why RAD domains exist is that they are merely for ad-blocker circumvention. Prior work has also assumed that serving similar ad-related content on several domains indicates evasion attempts [11, 56]. This section discusses some possible reasons for using RAD domains based on our observation in this study. Although it is challenging to precisely know the reasons that underlie the use of RAD domains, we manually inspected them and concluded four major reasons: customer isolation, localization, infrastructure changes or service rebranding, and ad-blocker evasion.

4.1.1 Customer isolation. Several advertisers use unique domains or subdomains for different customers, such that each of its ad domains only appears on one first-party domain. For example, we observed eight RAD domains owned by Reflektion, all of which were in the form of `*-prod.rfksrv.com`. We found that Reflektion assigned one domain for each customer exclusively, e.g., `259817494-prod.rfksrv.com` only appeared on the first-party domain `music-notes.com`. However, Reflektion might not intend to bypass ad blockers, because all its customers' ad domains were blocked by a wildcard rule, `||rfksrv.com^$third-party`. Some other cases were `podfdch.com` (`*.podfdch.com`, 67 RAD domains), Agile CRM (`*.agilecrm.com`, 20), and Insider (`*.api.useinsider.com`, 15).

4.1.2 Localization. Another possibly benign reason of RAD domain usage is localization. For instance, `counter.24log.it` was deemed a RAD domain of `counter.24log.ru` by our approach. By replacing the country TLD, we also found the German version (`24log.de`), the Spanish version (`24log.es`), and the English version (`24log.com`); all of them were blocked as of February 2021. Although they were all ad domains, their main goal seemed to be serving localized content.

4.1.3 Infrastructure changes or service re-branding. We also observed several cases that might be performing infrastructure changes or service rebranding. For example, `akamai-utility.rogersmedia.com` was a RAD domain of `utility.rogersmedia.com`. The former was observed only once (in November 2019) during our crawl, while the latter was observed several times between December 2015 and February 2021. Judging from its name, we concluded that the RAD domain might be created for testing the Akamai CDN. Another example is `www.afi-b.com`, which was a RAD domain of `www.affiliate-b.com`. The RAD domain first appeared in May 2017 and then was blocked in November 2020. After searching for information about this company, we found a press release issued in April 2017 announcing its service name change, from "Affiliate B" to "afb", and containing the new domain name `afi-b.com` [4]. Therefore, though ad blockers did not catch this RAD domain for more than three years, the reason of creating it seemed benign.

4.1.4 Ad-blocker evasion. Despite the existence of relatively benign reasons discussed above, we speculate that many of the RAD domains we found might be created for ad-blocker evasion. For example, some advertisers explicitly advised their customers to move ad domains to first-party subdomains for evading ad blockers, and their RAD domains indeed appeared and being blocked after their related ad domains. Nevertheless, for most cases, it is challenging to obtain concrete evidence of evasion attempts. We leave it as future work to identify domains that intend to evade ad blocking.

4.2 Recommendations

We now discuss general and pattern-specific detection against the four common domain-changing patterns (§3.1).

Our findings show that RAD domains were often left unblocked by today's ad blockers and survived for very long periods. We also found evidence that advertisers took advantage of the fundamental limitations of filter lists, creating domains to bypass ad blockers deliberately. Thus, it is more challenging to block such domains than regular ad domains. Generally, we recommend exploring automated

methods to detect RAD domains, thereby quickly spotting them and reducing their survival time. For example, one possibility is to integrate runtime information such as JavaScript execution traces and request traffic with content, URL, and ownership information, and apply machine-learning detection techniques [37, 38, 42] to complement our rule-based detection.

Moving to first-party subdomains. Ideally, we hope that first-party website owners can be transparent about their cooperation with advertisers, but they may hardly have incentives to do so. More practically, we recommend that the filter-list policies should not be lenient to first-party trackers but instead consider the actual information flow on the first-party domain. Also, we believe more studies are needed to weigh the advantages and disadvantages of providing ad blockers with access to DNS records. On the one hand, this allows ad-blockers to recover the actual destination following the DNS redirection (e.g., CNAME cloaking or simply pointing to the ad domains' IP addresses). On the other hand, granting more access to ad blockers may give rise to new types of privacy concerns.

Using revolving domains. As our findings show that it is promising to identify revolving domains via DNS records, we recommend that the filter-list maintainers and ad-blocker developers use DNS records for lightweight detection in addition to the general improvements discussed above. Although our findings show that the filter-list maintainers reacted fast to nearly half of the resolving domains we encountered, possibly through automatic scripts, we also point out that revolving domains can easily evade EasyList by leveraging outdated filter-list cache or deploying different domains per customer.

Changing subdomains. Some advertisers change subdomains of ad domains to bypass filter lists. Thus, to strengthen the defense against this domain-changing pattern, we recommend the filter list maintainers monitoring the subdomains of known ad domains.

Using CDN domains. We observed that some CDN providers used the same IP address or certificates for multiple customers. Thus, it is challenging to differentiate ad and regular domains merely by their owners. Therefore, for CDN domains, we recommend mainly using content-based or behavioral-based detection, as the ownership information may be misleading.

5 RELATED WORK

Many ad blockers use filter lists to identify ad domains, and thus their effectiveness heavily depends on the quality and completeness of the filter lists. Prior work has studied the issues of community-maintained filter lists and pointed out a handful of techniques to bypass them [11, 56]. Researchers have also analyzed specific evasion techniques, including CNAME cloaking [18, 44], exploiting cosmetic filters [57, 59], abusing WebSockets [12], and changing resource-serving domains or URLs [15, 58].

Among those touching upon advertisers' domain-changing behaviors, several studies [11, 56] have been limited to analyzing domains that have already been blocked, resulting in selection bias. Particularly, Alrizah et al. [11] found that many RAD domains were blocked shortly after the traffic commenced, implying that the EasyList community promptly and successfully detected them. Our findings, in contrast, suggest that RAD domains can survive for more than a year. In addition to the selection-bias issue, Snyder et al. [56] linked domains with identical file content, which did not work well for domains serving dynamic content or using contentless tracking methods such as tracking pixels. OmniCrawl [15] briefly presented a heuristic to discover RAD domains based on URL paths and TLS certificates. Inspired by these studies, our methods combine information extracted from DNS records, TLS certificates, file contents, and URL paths to search for RAD domains in the wild.

Besides using filter lists, recent work on ad blocking has also explored behavioral-based [16, 31] or content-based [21, 55] techniques to detect advertisements and tracers at runtime. Some focused on detecting tracking scripts using JavaScript execution traces [16, 33, 41], and some on detecting non-JavaScript ads or tracking contents by using DOM elements in HTML documents to train machine-learning classifiers [13, 37, 38, 42]. Our work can take advantage of these advances to improve the identification of similar-functionality domains. Of particular note is the work of Chen et al. [16], which built a tool to block privacy-and-security harming scripts at runtime and found 3,589 unique scripts not blocked by EasyList. They also presented a JavaScript-level evasion taxonomy (i.e., Moving, Inlining, Bundling, Common Code). By contrast, our work considered domain-level changes, and our analysis additionally incorporated temporal and ownership information.

Domain blocklists are commonly seen in the contexts of censorship [32] and malware detection [29]. In the former, censors build lists to block specific sites, and these sites try to bypass them. In the latter, antivirus software blocks access to specific command-and-control (C&C) servers, and malware tries to bypass and contact other C&C servers. One interesting future direction is to examine the domain-changing behaviors in these two fields, but our current methodology might not be directly applied. Our method to identify same-owner domains via DNS records and TLS certificates may be able to link re-hosted censored sites, but may fail to link C&C domains because C&C servers are usually compromised machines belonging to different victims.

6 CONCLUSION

To investigate advertisers' domain-changing behaviors, this work proposed methods to search for RAD domains produced by such behaviors and conducted a comprehensive analysis of their prevalence and impact. We found four common domain-changing patterns and discussed challenges to catch them using filter lists. Notably, the use of first-party subdomains to proxying third-party ads (i.e., the "moving to first-party subdomains" pattern) presents a significant challenge to ad blockers because of its extensive use, long survival time, and low blocking rate. Besides, it might introduce additional security issues that require further investigation. Our analysis also revealed that the non-blocked RAD domains extended the lifespan of ads or trackers for an average of 784.7 days. Moreover, 23.7% of the RAD domains exhibited privacy-intrusive behaviors, confirming their negative impact on ad blockers' privacy protection.

ACKNOWLEDGMENTS

This research was supported in part by the Ministry of Science and Technology of Taiwan under grants MOST 109-2636-E-002-021 and 110-2628-E-002-002 and by the National Science Foundation via grant CNS1704542.

REFERENCES

[1] [n.d.]. *EasyList Forum - Rules.* Retrieved September 20, 2021 from https://forums.lanik.us/rules#revolvingads

[2] [n.d.]. *Esprima.* Retrieved June 20, 2021 from https://esprima.org/

[3] 2015. *Issues with Yavli Advertising.* https://easylist.to/2015/08/19/issues-with-yavli-advertising.html [Online; accessed 20-September-2021].

[4] 2017. 株式会社フォーイットが提供するアフィリエイト・サービス サービス名リニューアルに伴うキャンペーン開催. Retrieved July 5, 2021 from https://www.for-it.co.jp/pressroom/pressrelease/20170414/

[5] 2017. *Create a tool/script to pickup revolving adservers.* Retrieved September 20, 2021 from https://issues.adblockplus.org/ticket/5323/

[6] European Commission 2018. *2018 reform of EU data protection rules.* European Commission. Retrieved October 16, 2020 from https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf

[7] AdBlock Support. 2021. *Introduction to Filter Lists.* Retrieved October 24, 2021 from https://help.getadblock.com/support/solutions/articles/6000066909-introduction-to-filter-lists

[8] Adguard Team. 2020. *Adguard - CNAME-cloaked trackers.* Retrieved October 16, 2020 from https://github.com/AdguardTeam/cname-trackers

[9] Adguard Team. 2020. *AdguardFilters - AdGuard Content Blocking Filters.* Retrieved October 16, 2020 from https://github.com/AdguardTeam/AdguardFilters

[10] aeris. 2020. *Address 1st-party tracker blocking.* Retrieved October 16, 2020 from https://github.com/uBlockOrigin/uBlock-issues/issues/780

[11] Mshabab Alrizah, Sencun Zhu, Xinyu Xing, and Gang Wang. 2019. Errors, Misunderstandings, and Attacks: Analyzing the Crowdsourcing Process of Ad-Blocking Systems. In *Proceedings of the Internet Measurement Conference* (Amsterdam, Netherlands) *(IMC '19)*. Association for Computing Machinery, New York, NY, USA, 230–244. https://doi.org/10.1145/3355369.3355588

[12] Muhammad Ahmad Bashir, Sajjad Arshad, Engin Kirda, William Robertson, and Christo Wilson. 2018. How Tracking Companies Circumvented Ad Blockers Using WebSockets. In *Proceedings of the Internet Measurement Conference 2018* (Boston, MA, USA) *(IMC '18)*. Association for Computing Machinery, New York, NY, USA, 471–477. https://doi.org/10.1145/3278532.3278573

[13] Jason Bau, Jonathan Mayer, Hristo Paskov, and John C Mitchell. 2013. A promising direction for web tracking countermeasures. *Proceedings of W2SP* (2013).

[14] Frank Cangialosi, Taejoong Chung, David Choffnes, Dave Levin, Bruce M. Maggs, Alan Mislove, and Christo Wilson. 2016. Measurement and Analysis of Private Key Sharing in the HTTPS Ecosystem. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS '16)*. Association for Computing Machinery, New York, NY, USA, 628–640. https://doi.org/10.1145/2976749.2978301

[15] Darion Cassel, Su-Chin Lin, Alessio Buraggina, William Wang, Andrew Zhang, Lujo Bauer, Hsu-Chun Hsiao, Limin Jia, and Timothy Libert. 2022. OmniCrawl: Comprehensive Measurement of Web Tracking With Real Desktop and Mobile Browsers. *Proceedings on Privacy Enhancing Technologies* 2022, 1 (Jan. 2022).

[16] Quan Chen, Peter Snyder, Ben Livshits, and Alexandros Kapravelos. 2021. Detecting Filter List Evasion with Event-Loop-Turn Granularity JavaScript Signatures. In *2021 IEEE Symposium on Security and Privacy (SP)*. 1715–1729. https://doi.org/10.1109/SP40001.2021.00007

[17] Conva Ventures Inc. 2020. Bypass ad-blockers with custom domains - Fathom Analytics. https://usefathom.com/blog/bypass-adblockers.

[18] Ha Dao and Kensuke Fukuda. 2020. A machine learning approach for detecting CNAME cloaking-based tracking on the Web. (2020), 1–6. https://doi.org/10.1109/GLOBECOM42002.2020.9322514

[19] Ha Dao, Johan Mazel, and Kensuke Fukuda. 2021. CNAME Cloaking-Based Tracking on the Web: Characterization, Detection, and Protection. *IEEE Transactions on Network and Service Management* 18, 3 (2021), 3873–3888. https://doi.org/10.1109/TNSM.2021.3072874

[20] Yana Dimova, Gunes Acar, Lukasz Olejnik, Wouter Joosen, and Tom Van Goethem. 2021. The CNAME of the Game: Large-scale analysis of dns-based tracking evasion. *Proceedings on Privacy Enhancing Technologies* 2021, 3 (2021), 394–412.

[21] Zainul Abi Din, Panagiotis Tigas, Samuel T. King, and Benjamin Livshits. 2020. PERCIVAL: Making In-Browser Perceptual Ad Blocking Practical with Deep Learning. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. USENIX Association, 387–400. https://www.usenix.org/conference/atc20/presentation/din

[22] Duck Duck Go, Inc. [n.d.]. *CATEGORIES - DuckDuckGo Tracker Radar.* Retrieved July 5, 2021 from https://github.com/duckduckgo/tracker-radar/blob/main/docs/CATEGORIES.md

[23] Duck Duck Go, Inc. [n.d.]. *DuckDuckGo Tracker Radar.* Retrieved July 5, 2021 from https://github.com/duckduckgo/tracker-radar

[24] EasyList contributors. [n.d.]. *EasyList - Policy.* Retrieved January 29, 2021 from https://easylist.to/pages/policy.html

[25] EasyList contributors. [n.d.]. *EasyList / EasyPrivacy / Fanboy Lists.* Retrieved October 20, 2020 from https://github.com/easylist/easylist

[26] EasyList contributors. [n.d.]. *EasyList / EasyPrivacy / Fanboy Lists Support.* Retrieved October 20, 2020 from https://github.com/easylist/easylist#support

[27] EasyList contributors. 2011. *What is acceptable first-party tracking?* Retrieved January 29, 2021 from https://easylist.to/2011/08/31/what-is-acceptable-first-party-tracking.html

[28] EasyList contributors. 2020. *EasyList - Overview.* Retrieved October 16, 2020 from https://easylist.to/

[29] Christopher M Frenz and Christian Diaz. 2017. Anti-ransomware guide. https://owasp.org/www-pdf-archive/Anti-RansomwareGuidev1-7.pdf. *Retrieved from owasp. org* (2017). [Online; accessed 20-September-2021].

[30] Google. 2020. *Public DNS | Google Developers.* Retrieved October 16, 2020 from https://developers.google.com/speed/public-dns

[31] David Gugelmann, Markus Happe, Bernhard Ager, and Vincent Lenders. 2015. An automated approach for complementing ad blockers' blacklists. *Proceedings on Privacy Enhancing Technologies* 2015, 2 (2015), 282–298.

[32] Joseph Lorenzo Hall, Michael D. Aaron, Stan Adams, Amelia Andersdotter, Ben Jones, and Nick Feamster. 2020. *A Survey of Worldwide Censorship Techniques.* Internet-Draft draft-irtf-pearg-censorship-04. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-irtf-pearg-censorship-04 Work in Progress.

[33] Muhammad Ikram, Hassan Jameel Asghar, Mohamed Ali Kaafar, Anirban Mahanti, and Balachandar Krishnamurthy. 2017. Towards seamless tracking-free web: Improved detection of trackers via one-class learning. *Proceedings on Privacy Enhancing Technologies* 2017, 1 (2017), 79–99.

[34] Internet Archive. 2020. *Wayback Machine.* Retrieved October 16, 2020 from http://web.archive.org/

[35] Internet Corporation for Assigned Names and Numbers. 2021. Temporary Specification for gTLD Registration Data - ICANN. Retrieved February 3, 2021 from https://www.icann.org/resources/pages/gtld-registration-data-specs-en/

[36] Umar Iqbal, Zubair Shafiq, and Zhiyun Qian. 2017. The Ad Wars: Retrospective Measurement and Analysis of Anti-Adblock Filter Lists. In *Proceedings of the 2017 Internet Measurement Conference* (London, United Kingdom) *(IMC '17)*. Association for Computing Machinery, New York, NY, USA, 171–183. https://doi.org/10.1145/3131365.3131387

[37] Umar Iqbal, Zubair Shafiq, Peter Snyder, Shitong Zhu, Zhiyun Qian, and Benjamin Livshits. 2018. Adgraph: A machine learning approach to automatic and effective adblocking. *arXiv preprint arXiv:1805.09155* 41 (2018).

[38] Umar Iqbal, Peter Snyder, Shitong Zhu, Benjamin Livshits, Zhiyun Qian, and Zubair Shafiq. 2020. Adgraph: A graph-based approach to ad and tracker blocking. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 763–776.

[39] John Wilander. 2020. *CNAME Cloaking and Bounce Tracking Defense.* Retrieved July 4, 2020 from https://webkit.org/blog/11338/cname-cloaking-and-bounce-tracking-defense/

[40] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* (1972).

[41] Andrew J. Kaizer and Minaxi Gupta. 2016. Towards Automatic Identification of JavaScript-Oriented Machine-Based Tracking. In *Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics* (New Orleans, Louisiana, USA) *(IWSPA '16)*. Association for Computing Machinery, New York, NY, USA, 33–40. https://doi.org/10.1145/2875475.2875479

[42] Amir Hossein Kargaran, Mohammad Sadegh Akhondzadeh, Mohammad Reza Heidarpour, Mohammad Hossein Manshaei, Kave Salamatian, and Masoud Nejad Sattary. 2020. On Detecting Hidden Third-Party Web Trackers with a Wide Dependency Chain Graph: A Representation Learning Approach. *arXiv preprint arXiv:2004.14826* (2020).

[43] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2019. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS 2019)*. https://doi.org/10.14722/ndss.2019.23386

[44] Arunesh Mathur, Jessica Vitak, Arvind Narayanan, and Marshini Chetty. 2018. Characterizing the Use of Browser-Based Blocking Extensions to Prevent Online Tracking. In *Proceedings of the Fourteenth USENIX Conference on Usable Privacy and Security* (Baltimore, MD, USA) *(SOUPS '18)*. USENIX Association, USA, 103–116.

[45] Mozilla and individual contributors. 2020. *dns - Mozilla | MDN.* Retrieved October 4, 2020 from https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/dns

[46] Mozilla Foundation. 2020. *Public Suffix List.* Retrieved October 20, 2020 from https://publicsuffix.org/

[47] NetApplications. 2020. *Browser market share.* Retrieved October 16, 2020 from https://netmarketshare.com/

[48] Paradot. 2021. *Add a Tracker Domain.* Retrieved September 22, 2021 from https://help.salesforce.com/articleView?id=sf.pardot_admin_add_tracker_domain.htm&type=5

[49] Peter Lowe. 2020. *Blocking with ad server and tracking server hostnames.* Retrieved October 16, 2020 from https://pgl.yoyo.org/adservers/index.php

[50] Plausible. 2021. *Serve the script from your domain as a first-party connection | Plausible docs.* Retrieved September 22, 2021 from https://plausible.io/docs/custom-domain

[51] Romain Cointepas, NextDNS Inc. 2020. *CNAME Cloaking, the dangerous disguise of third-party trackers | by Romain Cointepas | NextDNS | Medium.* Retrieved October 16, 2020 from https://medium.com/nextdns/cname-cloaking-the-dangerous-disguise-of-third-party-trackers-195205dc522a
[52] Nikita Savchenko. [n.d.]. *GitHub - dataunlocker/save-analytics-from-content-blockers: A proxy back end for Google Tag Manager & Google Analytics.* Retrieved September 20, 2020 from https://github.com/dataunlocker/save-analytics-from-content-blockers
[53] Sectigo Limited. 2020. *crt.sh | Certificate Search.* Retrieved October 16, 2020 from https://crt.sh/
[54] Segment.io, Inc. 2020. *Set up a custom domain proxy for Analytics.js -_ Segment Documentation.* Retrieved January 29, 2021 from https://segment.com/docs/connections/sources/catalog/libraries/website/javascript/custom-proxy/
[55] Alexander Sjösten, Peter Snyder, Antonio Pastor, Panagiotis Papadopoulos, and Benjamin Livshits. 2020. Filter list generation for underserved regions. In *Proceedings of The Web Conference 2020.* 1682–1692.
[56] Peter Snyder, Antoine Vastel, and Ben Livshits. 2020. Who Filters the Filters: Understanding the Growth, Usefulness and Efficiency of Crowdsourced Ad Blocking. In *Abstracts of the 2020 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems* (Boston, MA, USA) *(SIGMETRICS '20).* Association for Computing Machinery, New York, NY, USA, 75–76. https://doi.org/10.1145/3393691.3394228
[57] Florian Tramèr, Pascal Dupré, Gili Rusak, Giancarlo Pellegrino, and Dan Boneh. 2019. AdVersarial: Perceptual Ad Blocking Meets Adversarial Machine Learning. (2019), 2005–2021. https://doi.org/10.1145/3319535.3354222
[58] Phani Vadrevu and Roberto Perdisci. 2019. What You See is NOT What You Get: Discovering and Tracking Social Engineering Attack Campaigns. In *Proceedings of the Internet Measurement Conference* (Amsterdam, Netherlands) *(IMC '19).* Association for Computing Machinery, New York, NY, USA, 308–321. https://doi.org/10.1145/3355369.3355600
[59] Weihang Wang, Yunhui Zheng, Xinyu Xing, Yonghwi Kwon, Xiangyu Zhang, and Patrick Eugster. 2016. WebRanz: Web Page Randomization for Better Advertisement Delivery and Web-Bot Prevention. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (Seattle, WA, USA) *(FSE 2016).* Association for Computing Machinery, New York, NY, USA, 205–216. https://doi.org/10.1145/2950290.2950352
[60] Wikipedia contributors. 2021. *Tf–idf — Wikipedia, The Free Encyclopedia.* Retrieved February 3, 2021 from https://en.wikipedia.org/w/index.php?title=Tf%E2%80%93idf
[61] wordlists. 2017. *GitHub - xajkep/wordlists: Infosec Wordlists.* Retrieved January 29, 2021 from https://raw.githubusercontent.com/xajkep/wordlists/master/discovery/directory_only_one.small.txt
[62] George Kingsley Zipf. 2016. *Human behavior and the principle of least effort: An introduction to human ecology.* Ravenio Books.

A Limitations of our methodology

A.1 CDN and web-hosting services. Our methods for identifying same-owner domains assume that the entity controlling the hosting machine, the web content, or the TLS private key is the owner. However, this assumption may be false if the domain is hosted on a CDN. We observed that some CDN providers allocated the same IP address to multiple customers, or created so-called cruise-liner certificates [14]. Our manual inspection excluded false positives involving popular CDNs, but we might have missed cases hosted on CDN services that we did not recognize. Also, we observed that in some cases, the domain hosted on CDNs might be resolved to different data centers or different IPs of a data center, causing false negatives. However, on the other hand, linking all known IPs belonging to the same CDNs will create many false positives, and thus we did not take this approach.

A.2 URL path similarity. Our approach to collecting URL paths limited our findings to those captured by Wayback Machine. It therefore might be possible to improve the similarity measurement by actively traversing and collecting URL paths on websites directly. However, as well as imposing significant overheads on the websites in question, this would imply the loss of any historical perspective. Additionally, our URL path-similarity metric contains a threshold parameter affecting the tradeoff between false positives and false negatives. We conducted a manual inspection of our data to determine a suitable threshold value (as discussed in Appendix D.1). However, this threshold value may not be suitable for other datasets, for which additional tuning would be required.

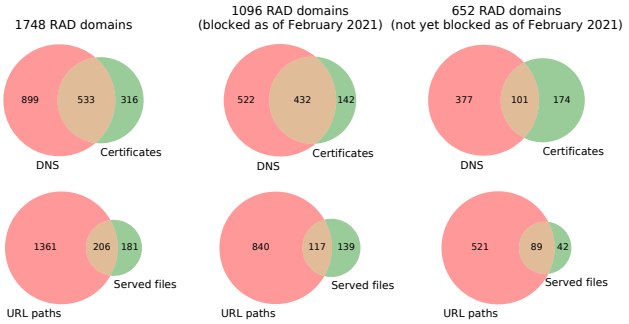

Figure 4: Venn diagram of RAD domains. Numbers represent the sizes of the exclusive power sets.

A.3 Functionality similarity via served files. Another method we used for evaluating functionality similarity is to calculate the Jaccard Similarity of either the SHA-256 hashes or the ASTs of file contents from two domains. This method, however, relies on the assumption that files served on two domains are the same if and only if two domains have similar functionality. As a result, this method may fail to link two similar-functionality domains with different file contents , such as subtle changes in images. In addition, sophisticated script obfuscation algorithms can easily bypass our AST-based method. However, we are unaware of any code similarity detection tools to date that can efficiently and accurately handle script obfuscation. We chose this method because it has proven effective and favors precision over recall, as shown in [16]. We reduced such false negatives by considering their path similarity as well. On the other hand, two unrelated domains may be considered having similar functionality if they serve some well-known files such as jQuery scripts. We reduced such false positives by setting the Jaccard Similarity threshold. Moreover, we relied on the same-owner requirement and manual validation to further reduce the chance of having falsely linked domains.

B RAD domain relationship

B.1 Ownership. To ascertain if there was any redundancy between our two methods of linking domains with the same owner (i.e., DNS records and TLS certificates), we grouped RAD domains by the method that could be used to discover them, and found that no group fully overlapped with another. Figure 4's Venn diagram shows how many of the RAD domains we identified could be linked via each of our methods, and clearly illustrates that these two methods complement each other, and each has its own merits.

B.2 Similar-functionality. To determine if there was any overlap between our methods of linking domains with similar functionality (i.e., URL paths and served files), we grouped RAD domains according to how they were discovered, and found little overlap. In Figure 4, the Venn diagram demonstrates how many of the RAD domains we identified could be linked to each other using our

Table 2: Categories of Purposes from Tracker Radar. Adapted from [22].

Category	Description
Ad Motivated Tracking	Trackers related to ads, including user targeting, header bidding, demographic collection, etc.
Advertising	Trackers related to ads.
Analytics	Trackers for analytics, including market analytics, website analytics, etc.
Audience Measurement	Trackers for analytics, but focus more on demographics, behavior sets, and specific actions.
Third-Party Analytics Marketing	Trackers related to third-party analytics systems for marketing, usually marketing attribution or funnel management.
Embedded Content	The domain is used to embed content, such as Youtube, Vimeo, widgets, etc.
Action Pixels	Trackers that collect user specific events.
Ad Fraud	Trackers to prevent ad fraud.
Social Sharing Buttons (Social - Share)	Third-party SDK powered social sharing buttons, such as Facebook's share buttons.
Session Replay	Recording visitors' behaviors. Often records much more user behaviors than typical analytics libraries.

Table 3: Numbers of RAD domains exhibiting common patterns and statistics on their survival times.

Pattern	RAD Domain		Survival Times of Block Domains (days)	
	All	Blocked	Mean	Median
Moving to first-party subdomains	305	135	436.0	196
Using revolving domains	222	154	95.9	37
Changing subdomains	627	524	419.2	224
Using CDN domains	167	34	249.1	151
Domains having at least one pattern	1,217	790	336.4	152.5
All RAD domains	1,748	1,096	410.5	195.5

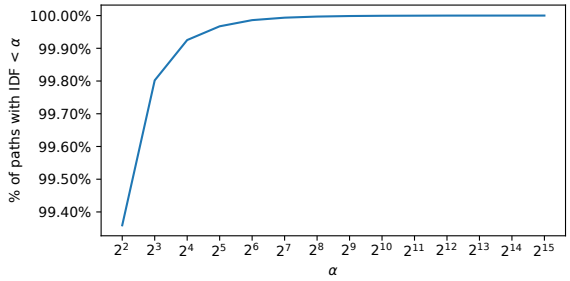

Figure 5: Cumulative distribution function of paths with Inverse Path Frequency less than α domains.

two methods; this shows how these two methods complement one another and have strengths and weaknesses of their own.

We then looked into the similar paths and the set of served files that link RAD domains. For most URL-path-linked domains, the length of similar paths was surprisingly short, with only 14.71 characters and 2.71 path segments [5] on average, for we expected that the paths should be long enough to be unique. We inspected the short similar paths and found that most of them were indeed unique enough to be identified, such as `/vast-im.js` and `/wrapperMessagingWithoutDetection.js`. Meanwhile, we also observed several long URL paths, such as `/mn9117912/ilvpm003y/oln/786/vqu768kypc01r` and `/bultykh/ipp24/7/bazinga`. This confirms our assumption that similar paths indicate similar functionalities.

For the served-file-linked domains, we observed that most of them have small sets of files, with 1.65 files on average. Moreover, among 387 domains linked via served files, 335 of them served only one file, including 57 revolving domains. The majority of those that served small numbers of files and were linkable only via the served files had randomized or obfuscated file paths. For example, we encountered an ad domain `qakdki.com` with the path `/i2dl71/921livp0m30yh8q/867vuq/768kypph1zo.php`, and its RAD domain `zesdmn.com` with the path `/5ka1l7291/vli0mp30yq8h687uqv/876kypyv.php`. Both URLs pointed to the same file and exchanging the paths between the two domains also worked. Furthermore, some ad domains seemed to serve the identical files across all paths. For example, we encountered `kont-news.com/mKJg.js` during the crawl, and by manual inspection, the ad server served the same file for all four-character-long file names, such as `kont-news.com/abcd.js`. Our suspicion is that this method was used to circumvent blocking by the filter lists via URL paths. It justifies the need for the served-file relationship, since the similar-path relationship may miss these cases.

[5]Note that `/abc/def.js` has three path segments: `/`, `abc/`, `def.js`

C Prevalence of browser fingerprinting

As we presented in Section 3.3, a high percentage of ad domains were tracking users. We also analyzed how many RAD domains served scripts for browser fingerprinting, one of the most privacy-intrusive techniques to achieve tracking.

We utilized the Tracker Radar dataset [23] again to determine the likelihood that a domain was fingerprinting users. The likelihood is categorized from the lowest to the highest as: Certainly Not, Not Obviously, Possibly, Certainly. "Certainly Not" means the domain does not use browser APIs, and "Certainly" means the domain uses browser APIs excessively and almost certainly for tracking purposes. Similar to Section 3.3, we utilized the RAD relationships to fill in missing information. If a RAD domain had no likelihood data, we assigned it the lowest level among all of its related ad domains, and same applies to the ad domains. In the end, we labeled the likelihood level of 2,520 ad domains and 395 RAD domains, 290 of which are inferred from their related ad domains.

Our analysis shows that 253 (64.1%) RAD domains were probably not doing browser fingerprinting. However, 142 (35.9%) RAD domains were possibly or certainly serving fingerprinting scripts. This observation shows that some advertisers use RAD domains to serve privacy-intrusive scripts. However, we did not see statistical differences in the distribution between ad domains and RAD domains.

D Parameter selection

D.1 URL similarity. We mentioned in Section 2.1.2 that the similarity of URL paths is an indicator of similar functionalities. In this section, we describe our parameter selection process for α, as it may affect the precision and recall. To choose the parameter α, we first computed the cumulative distribution of paths as shown in

Figure 5. As the trend follows Zipf's law [62], the number of paths starts to decrease dramatically when $\alpha \geq 2^3$. Thus, we next manually inspected the paths where $2^3 \leq \alpha \leq 2^{15}$. When $\alpha > 2^{11}$, and saw that common WordPress paths `wp-includes` were included. When $\alpha = 2^{11}$, unwanted paths like `uploads/` and `scripts/` were included. We empirically set α to 2^{10}, though `stylesheets/` was included, it seems to be reliable under $\alpha = 10$. According to our manual validation, overall, our heuristic yields a reasonable true positive rate.

Since α may affect the precision and recall, ideally, we should find a α which maximizes the F1 score. However, as there is no ground truth of RAD domains and manually labeling a large set of data is infeasible, we choose a relatively conservative α to strike a balance. Estimating the F1 score and fine-tuning α is left as future work.

D.2 Served files. We randomly sampled 100 pairs of domains known to be the RAD domains of each other, and sampled another 200 domains and pair them randomly. The distribution of the Jaccard similarities of the two populations is extreme, for most are either close to zero or one. We empirically chose the threshold at 0.7.