Knockin' on Trackers' Door: Large-Scale Automatic Analysis of Web Tracking

Iskander Sanchez-Rola and Igor Santos

DeustoTech, University of Deusto
{iskander.sanchez,isantos}@deusto.es

Abstract. In this paper, we present the first generic large-scale analysis of different known and unknown web tracking scripts on the Internet to understand its current ecosystem and their behavior. To this end, we implemented TrackingInspector the first automatic method capable of detecting generically different types of web tracking scripts. This method automatically retrieves the existing scripts from a website and, through code similarity and machine learning, detects modifications of known tracking scripts and discovers unknown tracking script candidates.
TrackingInspector analyzed the Alexa top 1M websites, computing the web tracking prevalence and its ecosystem, as well as the influence of hosting, website category, and website reputation. More than 90% websites performed some sort of tracking and more than 50% scripts were used for web tracking. Over 2,000,000 versions of known tracking scripts were found. We discovered several script renaming techniques used to avoid blacklists, performing a comprehensive analysis of them. 5,500,000 completely unknown likely tracking scripts were found, including more than 700 new different potential device fingerprinting unique scripts. Our system also automatically detected the fingerprinting behavior of a previously reported targeted *fingerprinting-driven malware* campaign in two different websites not previously documented.

Keywords: device fingerprinting; privacy; web tracking

1 Introduction

Web tracking is a common practice on the Internet to gather user browsing data for different tasks such as advertisement, personalization, analytics, and identity checking. Despite the fact that web tracking poses a threat to users' privacy and anonymity, it is not considered harmful by itself. Indeed, web advertising companies defend web tracking as a fundamental component of web economy [48]. However, recent reports [44, 51] found two different targeted malware campaigns that used a fingerprinting phase to increase the success rate and hide them.

Recent work studied both stateful [32, 41, 43] and stateless [2, 3] tracking, raising a general concern about its prevalence in the web. These studies provided a better understanding of a particular subset of web tracking techniques but they were not devoted to fully understand and to generically discover web tracking scripts. Another recent work [17], combines some of the previous approaches and ideas to analyze and study both stateful and stateless web tracking. However, due to the nature of these proposed approaches, only concrete web tracking techniques are analyzed and, thereby, a generic web tracking analysis cannot be performed with these methods.

Given this background, we present the first large-scale analysis of generic web tracking scripts. Due to the limitations of current solutions, we build our own tracking analysis tool called TrackingInspector. In contrast to existing solutions, based either on blacklists or static rules, this tool is based on code similarity and machine learning. TrackingInspector automatically detects known tracking script variations and also identifies likely unknown tracking script candidates. We use TrackingInspector to analyze the Alexa top 1M sites [46] to answer the following research questions: (i) how widespread is web tracking on the Internet?, (ii) what is the current ecosystem in web tracking provision and deployment on the Internet?, (iii) can current blacklisting solutions limit or block most web tracking on the Internet?, and (iv) can TrackingInspector discover new web tracking scripts? To what extent?

The major contributions and findings of this paper are three. First, we performed the first large-scale study of generic web tracking. More than 90% of the websites performed tracking and more than 50% of the scripts exhibited a tracking behavior. Second, we present TrackingInspector, the first tool to automatically detect generic tracking scripts through code similarity and machine learning. The results show its ability to automatically detect known tracking scripts and their modifications, and to discover potentially unknown tracking. Our method was able to detect more than 2M known tracking script versions whereas current blacklisting solutions were only able to detect 64.65% of them in the best scenario. This indicates that many variations of tracking scripts are bypassing current solutions. Therefore, we studied these hiding techniques, finding several *script renaming techniques*. Finally, more than 5.5M not previously reported scripts exhibited a tracking behavior. These scripts include over 700 new unique potential device fingerprinting scripts: more than 400 performing canvas fingerprinting, more than 200 performing font probing, and more than 50 exhibiting both. TrackingInspector also automatically detected a previously reported targeted *fingerprinting-driven malware* campaign exhibiting fingerprinting behavior, present in two websites not reported as infected.

2 Tracking Analysis & Detection

2.1 General Description

It is important to remark that the definition of web tracking has been a controversial debate due to the different existing tracking types. In this work, we are going to use Ghostery's[1] definition, which is also used by many other security

[1] `https://www.ghostery.com/submit-a-tracker/`

and privacy companies: *"Trackers (also commonly referred to as tags) are snippets of code that send and receive information about you to companies to track and analyze your behavior, deliver ads, connect social media, provide comment sections, and more."*

Current Solutions We evaluated the following solutions for tracking detection: blacklisting (e.g., *EasyPrivacy* [4] and *Ghostery* [9]), the EFF's tool *Privacy Badger* [24] based on simple heuristics, *FPDetective* [3], and *OpenWPM* [18].

Blacklisting tools rely on blacklisted script names, URLs, and domains. Although these methods detect the known tracking scripts and domains, they fail to detect simple variations such as renaming the script or modifying the domain where the scripts are hosted. Besides, they may incorrectly block scripts devoid to tracking but named with the same script name as one within the blacklist.

Heuristics used in the *Privacy Badger* plugin block third-party cookies. This approach raises false positives and only focuses on cookies. Tracking adopts many different forms not covered by this tool.

FPDetective and *OpenWPM* are tracking analysis frameworks based on blacklisting and/or rules to detect tracking. These frameworks solve the main limitations of blacklisting. However, as these techniques are based on predefined rules, the tracking script can be modified with methods that bypass the defined criteria (see §5 for some examples).

TrackingInspector Solution Our solution is composed of three components:

- A *Crawler* (§2.2) to retrieve web scripts and content.
- A *Script Database* (§2.3) with known tracking scripts.
- A *Text-based Analyzer* (§2.4) to analyze the scripts using the *Script Database*. It detects both known (or versions) and potentially unknown tracking.

TrackingInspector starts by downloading scripts through the *Crawler*. The *Crawler* waits until every script is downloaded; including third-party, first-party, or HTML-embedded scripts. Then, *Text-based Analyzer* analyzes them using its two components:

1. *Known Tracking Analysis* measures the similarity of each script with the ones stored in the *Script Database*. The script can be a version of a known tracking script or not. This analyzer is intended to be an end-user solution.
2. When no match is found, the *Unknown Tracking Analysis* inspects the script using a machine-learning algorithm trained with the *Script Database* to detect likely unknown tracking scripts. This component is devoted to find new tracking scripts to be added to the *Script Database*.

Dynamic approaches monitor the web behavior during browsing and compare it with specific rule sets for concrete types of tracking [3]. Instead, TrackingInspector uses the *Crawler* to dynamically retrieve the scripts of the site and then, the scripts are analyzed by the *Text-based Analyzer*.

Script Representation We tested two different approaches: Abstract Syntax Trees (ASTs) and Bag of Words (BOW). While ASTs represented the specific syntax of the functions within the code, the BOW approach captures the token frequencies to model the script. In our preliminary tests, the BOW text-categorization approach behaved better to detect generic tracking behaviors. AST approaches represent the code syntax strictly taking into account also the script structure, while BOW just represent the usage of tokens. Hence, ASTs are worse when dealing with script modifications or new scripts than the standard BOW model. BOW models the scripts using a Vector Space Model (VSM), which represents them as vectors in a multidimensional space whose dimensions represent the possible tokens within the scripts. Scripts are represented as a vector of frequencies of tokens. Since our goal is not to capture the behavior but to detect any type of tracking, other approaches may overfit and fail to detect modifications or new web tracking scripts.

BOW has been used for a broad range of web security problems such as detection of drive-by-download attacks [42], to measure the impact of different models in malware detection [8], or vulnerability detection [56].

2.2 Crawler

Overview The *Crawler* component automatically retrieves every JavaScript file statically or dynamically loaded in a website. The *Crawler* is based on *PhantomJS* [27], a well-known headless browser. To avoid the detection of the automated browsing when using headless browsers, we adapted existing advanced hiding methods [47] to disguise the browser as a common one. With this adaptation, our *Crawler* is capable of performing transparent and exhaustive browsing. The *Crawler* also deals with obfuscation, and cleans the generated caches and cookies after each website inspection.

Methodology The *Crawler* starts visiting the frontpage of the site under inspection. It saves all the downloaded scripts, taking into consideration if different scripts have the same name or if they are downloaded multiple times. To retrieve them, instead of waiting a fixed time and gathering them, like previous work on fingerprinting detection [2,3], we analyzed the different script generation behaviors in 250,000 random websites within the Alexa top 1M websites to define a methodology for web analysis and set the adequate times to retrieve every script.

The resulting methodology starts with the *Crawler* waiting until all frames in the website are loaded with a fixed maximum time of 60 seconds. If all frames are loaded before the 60 seconds are elapsed, the *Crawler* waits 15 seconds more (but never more than the maximum time). We added this second waiting time because in several cases additional scripts were downloaded after every frame had already been loaded because other scripts may have called them. If scripts are downloaded during this extra time window, the *Crawler* will wait until the remaining of the 60 seconds are elapsed. These time frames were selected taking into account the possible redirections within the website or its frames.

Then, the *Crawler* retrieves the resulting HTML code, with all the generated modifications, and gathers the HTML-embedded scripts. Then, the *Crawler* starts a deobfuscation phase and finally, cleans duplicate scripts, files that are not actually JavaScript, or empty files.

De-Obfuscation We implemented a deobfuscator using *JSBeautifier* [33], a well-known deobfuscator as a starting point. Using the techniques implemented in this tool, our deobfuscator tries to unravel the original code, checking in each iteration whether or not the script has been deobfuscated with the specified metrics. Therefore, we can retrieve the original code conserving variable names and structure. Even if many approaches to obfuscate code exist, the analysis of Curtsinger et al. [12] found that the simple `eval` unfolding is the most commonly used, which is included in our deobfucator along with others.

In this way, we can deal with multiple layers and multiple known techniques of obfuscation. In fact, during our work, we found some obfuscators that use others iteratively to perform multiple layer obfuscation. When it is deobfuscated, our method performs an additional step to deal with one obfuscator with a particular behavior. This particular one, when the free version is used, places the original code in an escaped string within the code while the rest of the code is used to call functions in the string and also performs a callback function to notify the authors that this script has been executed. Our method retrieves the code stored in the string and unescapes it, discarding its additional code.

Limitations This approach may also have its shortcomings. In particular, a dedicated tracker can use more advanced obfuscation to bypass current *JSBeautifier*-based deobfuscation pass.

2.3 Script Database

It stores both known tracking and non-tracking scripts. For the non-tracking scripts, we downloaded scripts from open-source projects and from randomly accessed websites (that did not belong to the Alexa top 1M sites), manually verified afterwards.

Data Sources To generate the tracking scripts dataset, we retrieved scripts on the following sources:

- **Blacklists:** We used *EasyPrivacy*, *Kaspersky Tracking List (ABBL)* [30], *ABINE* [1], the tracking version of *AdGuard* [5], the tracking list *FanBoy* [19], and the *Tracking Detection System (TDS)* by Rob van Eijk [16]. These lists were selected because they include scripts and not just domains. We omitted blacklisted domains because our goal is to detect tracking scripts rather than domains. However, this information was used in the large-scale analysis (along with other 6 tracking domain blacklists that will be detailed in §3) to compare the results and findings of TrackingInspector. Some of these lists included a whitelist composed of tracking scripts that are not considered harmful. Since our goal is to detect tracking behavior, we also included this type of scripts.

- **Open-source Tracking Projects:** We retrieved several open-source tracking projects: *BeaverBird* [45], *FingerPrintJS* [54], and *Evercookie* [29].
- **Academic Papers:** We also included in our tracking *Script Database* the scripts found in [2, 3].

We processed the list of potential scripts and stored the scripts whose complete URL was available. When the scripts were not available, we tried to download them through `archive.org`, removing all the service-related text and code afterwards. When script names were only available, we searched it using several code searchers (e.g., *meanpath* [38], *NerdyData* [13], *FileWatcher* [21]), or common search engines (e.g., *Google* and *Bing*). Then, we manually checked each script to determine whether they performed tracking or not.

Configuration Since some of the scripts were obfuscated, we deobfuscated them as described in §2.2. Then, we removed duplicates or different versions of scripts. To this end, we modeled the code using the representation detailed in §2.1 and computed the cosine similarity of each script within the *Script Database*.

To set a threshold for the detection of versions of original scripts, we conducted an empirical validation over the scripts and selected 85% because it was the lowest one (more coverage), raising no false positives. Then, we added the original scripts to the *Script Database* and also a small number of script versions that presented new functionalities to the original. After this process, 957 original tracking scripts were stored in our *Script Database*. We randomly selected 957 non-tracking scripts from the aforementioned sources.

2.4 Text-based Analyzer

Overview The *Text-based Analyzer* is responsible for the detection of tracking scripts. This component is divided in two different sub-components: a (i) *Known Tracking Analysis*, responsible for the detection of versions, or modifications of known scripts stored in the *Script Database* and a (ii) *Unknown Tracking Analysis*, whose goal is to automatically identify tracking script candidates.

Text-based Web Tracking Detection Although the goals of known and unknown approaches are different, they both represent scripts using the BOW approach. As in the *Script Database*, the scripts are modeled through a VSM, composed of the terms within the scripts. *Text-based Analyzer* represents each script as a sequence of each term frequencies, using the well-known *Term Frequency [34] – Inverse Document Frequency* (TF–IDF) [49] weighting schema.

Known Tracking Analysis detects versions or modifications of currently known tracking stored in the *Script Database*. This component computes the cosine similarity of the script under inspection with known tracking. The cosine of the angle formed by the vector representation of the two scripts is the similarity — if they are totally equal the angle will be zero and their similarity 1, while when the angles are orthogonal and hence they do not share any token, their similarity score will be 0. When the empirically computed threshold of 85% similarity is surpassed, the script is flagged as a known tracking script version.

Unknown Tracking Analysis is based on supervised machine learning. The features used for the machine algorithm are the tokens in the BOW model. Machine learning develops algorithms to learn automatically behaviors from data. Since, in our particular case, data is labeled (i.e., tracking and non-tracking), supervised machine learning algorithms were selected for this task. These algorithms have been extensively used in web security tasks such as detection of malicious websites [7] or malicious JavaScript code [11]. We use the *Script Database* to perform a 10-fold stratified cross-validation experimental evaluation with several well-known machine-learning methods (e.g., Naive Bayes, Bayesian Networks, C4.5, SVMs, and Random Forest) to decide which classifier to use. The best classifier was *Random Forest* [28] configured with 950 *Random Trees*. This ensemble method for classification creates several decisions trees at training and chooses the classification based on the mode or mean of their partial classifications. In the training phase, the bagging technique is used to create the weak random tree learners. To build the aggregate, a similar but more general method is used to select the different high-level splits, also known as feature bagging.

Evaluation We evaluated the performance of these two components of the *Text-based Analyzer* in terms of tracking script detection, prior to their usage in our large-scale analysis:

– **Known Tracking Analyzer:** Using 85% similarity threshold, did not report any false positives in our tests, being able to detect any version of known tracking at least 85% similar to samples in the *Script Database*. Therefore, we believe that this component should be used in an end-user environment as a replacement of blacklisting techniques. We will compare it with blacklisting solutions in §3.
– **Unknown Tracking Analyzer:** We consider scripts not detected by the *Known Tracking Analyzer* as *previously unknown*. During the cross-validation evaluation, this component achieved an area under the ROC curve of 0.982, a true positive rate of 94.4%, and a F-measure (the harmonic mean of precision and recall) of 0.935. This method is the first one in the literature able to detect previously unseen tracking and devoted to the discovery of new tracking rather than using it in an end-user environment. This discovery of potentially tracking acts as a filter for a manual inspection to include actual tracking in the *Script Database*. To further evaluate the *Unknown Tracking Analysis* component, an additional evaluation of the method will be performed through a manual inspection of tracking scripts in the wild in §3.4.

To sum up, both components comply with our initial requirements to be used in the large-scale measurement and analysis.

3 Large-Scale Analysis

3.1 Preliminaries

The goal of this analysis is to answer the next research questions: (i) *how widespread is web tracking on the Internet?*, (ii) *what is the current ecosystem in web tracking provision and deployment on the Internet?*, (iii) *can current blacklisting solutions limit or block most web tracking on the Internet?* and (iv) *Can* TrackingInspector *discover new web tracking scripts and to what extent?*

The *Crawler* retrieved the scripts within the Alexa top 1M and the *Text-based Analyzer* inspected them. When downloading scripts from a removed site, we searched it through `archive.org`. Since `archive.org` adds code and also files to the website, we made a cleaning process. 3.67% of the websites were not accessed because its access was restricted (401 and 403), not accessible, or were impossible to find in `archive.org`.

Some of the scripts flagged as likely unknown tracking scripts by the *Unknown Tracking Analyzer* may be only unknown by TrackingInspector but known by existing domain blacklisting tools. To discriminate between these two cases, we used the blacklisting tracking detection tools *EasyPrivacy*, *Kaspersky Tracking List (ABBL)*, *ABINE*, the tracking version of *AdGuard*, the tracking list of *FanBoy*, the *Tracking Detection System (TDS)*, *Ghostery*, *Privacy Badger*, *Disconnect* [14], *Truste* [53], *Privacy Choice* [50], and *Web of Trust* [52] (the domains classified there as the category 301 - online tracking).

We gathered data about the hosted website and the top-level domains where the scripts were downloaded. We also analyzed the domains, because the hosted scripts in domains influence the trust of the sites [40]. To correctly retrieve the top-level domain names, we used the `effective_tld_names.dat` by Mozilla [25]. Next, we extracted the country, the ISP, the associated ASs, and the web category of the domains. To determine the category, we used three services: *Cloudacl* [10], *Blocksi* [6], and *Fortiguard* [23]. Their category names are similar, and, after a normalization process, 78 category names were fixed. We also analyzed the reputation of the websites in the *webutation* service [55], that uses users' feedback, comments, and also different analyses such as *Google Safe Browsing*, *Norton Antivirus* or `phistank.com`.

For the sake of clarity, we define the terminology that we will be using:

- **Known Tracking Scripts:** We call known tracking scripts to those in the range between 85% and 100% similarity to the ones in the database.
- **Unknown Tracking Scripts:** Scripts that are less than 85% similar to the ones in the script database. These scripts can be either new scripts from scratch or versions of known ones modified enough to be considered *new*. Examples of unknown tracking scripts may include new implementations, added functionalities, or simply scripts never reported and thus never blacklisted. We will also refer to them as *tracking script candidates* or *likely tracking scripts* as the unknown text-analyzer has a small error percentage. We also distinguish between two sub-categories.

Table 1: Tracking and non tracking behavior prevalence in scripts.

Type	**# Scripts**	**% Scripts**
Tracking	11,984,469	57.15%
Non tracking	8,985,457	42.85%
TOTAL	*20,969,926*	*100.00%*

Table 2: Tracking and non tracking behavior prevalence in websites. % W. S. stands for the percentage of websites, considering only websites with scripts. % W. represents the percentage considering every website.

Type	**% W. S.**	**% W.**	**# Websites**
Tracking	97.58%	92.89%	894,779
Non tracking	2.42%	2.31%	22,220
No scripts	N/A	4.80%	46,277
Number of websites with scripts			*916,999*
Total number of websites			*963,276*

- *Unknown Blacklisted Tracking Scripts:* Unknown scripts whose hosting domain is blacklisted but the script is not.
- *Completely Unknown Tracking Scripts:* Unknown scripts whose hosting domains have not been blacklisted.

We also classify them as: (i) *Original*, if in the script database; (ii) *Unique*, different compared by hash; and (iii) *Version/Sample* each download of a script.

3.2 Tracking Ecosystem

General Overview 20,969,926 script samples (tracking and non-tracking) were downloaded from the the Alexa top 1M websites. Impressively, nearly 60% of them were flagged as tracking (see Table 1). In other words, more than half the script functionality in the web is potentially devoted to track users. There were 46,277 websites with no scripts at all. Hence, we measure the tracking prevalence percentage both in every website and in websites with scripts (see Table 2), finding that nearly every website performed tracking.

Only a 20% of the tracking samples were known, whereas the unknown samples were the stunning majority. However, using the previously omitted domain blacklists, 41.11% of these scripts were in blacklisted domains (see Table 3), being unknown blacklisted scripts. The number of samples where neither the script nor the domain were blacklisted, was an impressive 46.90%. Tracking scripts were downloaded from 891,873 different domains.

Website Demographics For a better understanding, we analyzed different aspects and tracking prevalence. To find the relevance of tracking in each analyzed

Table 3: Detected tracking script distribution. *Domains* refer to top-level domains where the scripts are downloaded. *Unknown (blacklisted)* refers to likely tracking script candidates unknown in the Script Database but whose domain is blacklisted.

Type	**# Scripts**	**# Domains**
Known	2,439,835	540,369
Unknown (blacklisted)	3,923,615	7,455
Unknown	5,621,019	841,425
TOTAL tracking	*11,984,469*	*891,873*

feature (website, category/*webutation*, and country/network entity origin in domains), we ran several preliminary tests computing the differences of tracking ratios per website to find which ratio could discriminate between the features. The results showed that computing the number of websites with only tracking scripts per each studied feature eased the discrimination, while the number of websites with some tracking and the number of scripts per featured showed the overall behavior.

We computed the following ratios: (i) % of tracking scripts per script in the category, (ii) % of websites performing any type of tracking per website in the category, and (iii) % percentage of websites with only tracking scripts per website in each category. The website categories with the highest tracking percentage were *personal websites*, *hacking*, *spyware and adware*, *social networks*, or *peer to peer*. The top categories with only tracking were *malicious*, *questionable*, *unknown*, and *websites with adult content*. Despite it cannot be used to discriminate the maliciousness or greyness, *malicious* or *grey* sites tend to only include tracking.

The relation between websites with some or only tracking scripts with *webutation* hinted that the presence of only tracking affects the reputation of a website. 15.41% of the websites in the *Red* category and 15.31% in the *Yellow* category only used tracking, while only 6.45% and 5.95% of the *Grey* and *Green* categories did. Since users are not usually aware of web tracking, we believe that these results indicate that sites perceived as *bad* by users have a higher ratio of tracking than non-tracking, similar to what happened with categories.

Domain Demographics We also measured domains hosting tracking, non-tracking, and both tracking and non-tracking scripts (see Table 4). We analyzed domains to understand the provision of web tracking. In fact, previous work found the correlation between the scripts in domains and their nature [40].

We discovered that, similarly to what happened to websites using tracking scripts, domains usually host both web tracking and non-tracking scripts. However, the percentage of them hosting solely tracking scripts is not negligible (10.54%). Regarding the relation of countries with their domains, several small countries surprisingly hosted only tracking. Likewise, we found cases of either

Table 4: Domain distribution with regards to tracking. *Only Tracking* represents the top-level domains that only contain tracking scripts, whereas *Only non tracking* represents top-level domains with only non-tracking scripts. *Tracking & non tracking* represent the top-level domains that contain both tracking and non-tracking scripts.

Type	**# Domains**
Only tracking	98,359
Only non tracking	41,640
Tracking & non tracking	793,515
TOTAL	*933,514*

AS owners, ASNs, or ISPs whose domains were only used to host tracking. Nevertheless, given the small number of domains, we consider them irrelevant.

As we did with sites, we studied the correlation between the *webutation* of the domain and its hosting of tracking scripts. We found that, as happened with websites, the presence of only tracking in domains affected the reputation: *Yellow* and *Red* represented 22.87% and 23.71% of the domains hosting only tracking scripts while *Grey* and *Green* categories only contained 11.23% and 9.44%.

3.3 Analysis of Known Tracking

To compare the detection capabilities of TrackingInspector with current tracking blockers, we measured the number of known script samples that blacklisting solutions would have blocked. From all the methods, we chose script name and domain blacklisting as the baseline because they provided a broader detection compared to the other alternatives. Another possible solution not used in tracking detection but used in other domains was also compared: code hashing.

Results show that script and domain blacklisting captured 43.80% and 33.84% of the known tracking script versions, respectively. Combined blacklisting solutions blocked the 64.65% of the known tracking scripts while code hashing only would had captured 2.04% of the samples. These results show that current anti-tracking solutions are clearly not enough, not only to fight against completely unknown tracking scripts, but also against modified known tracking scripts.

Moreover, we measured the prevalence of versions of the tracking scripts in the *Script Database*. *Google* related scripts were the most popular: 60.90% of the samples correspond to their scripts, including 29.27% of samples with analytics capabilities. Among other scripts we can find: 20.92% regarding advertisement (*33Across*, *Pzyche*, and *QuantCast*), 3.49% from analytics (*Yandex Metrica* and *comScore*), and 2.00% social analytics samples (*FlCounter* and *Pinterest*). These ones were used by the 84.02% of the websites with scripts.

540,369 top-level domains hosted the known tracking samples (the most popular one was `google-analytics.com`). Their scripts were present in 63.14% of the websites with scripts. The rest of the domains belonged to Google or to

Table 5: Unknown tracking downloaded from blacklisted domains prevalence in websites.

# websites	646,428
– % in websites with tracking	72.24%
– % in websites (considering sites with scripts)	70.49%

Table 6: 10 most popular blacklisted domains hosting unknown tracking scripts.

Domain	**# Websites**
`facebook.com`	177,443
`akamaihd.net`	176,616
`googlesyndication.com`	166,469
`google.com`	156,214
`twitter.com`	120,843
`gstatic.com`	114,153
`facebook.net`	86,299
`googleusercontent.com`	86,023
`googleadservices.com`	83,676
`ytimg.com`	72,571

well-known advertisement services. The hosting of known tracking scripts follows a long-tail distribution with a small number of domains (or companies) representing the majority of script downloads.

3.4 Analysis of Unknown Tracking

Unknown Tracking Analysis In-the-wild Evaluation TrackingInspector flagged more than 9.5M scripts as unknown tracking and more than 5M were not previously known by any blacklist. Albeit we already performed a 10-fold cross validation, we also performed an in-the-wild manual validation.

To this end, we extracted a statistically significant random sample from all the scripts flagged as unknown tracking. The sample was composed of 273 scripts, representing a 90% confidence level and a ±5% confidence interval. According to statistical sampling [22], confidence level measures the probability of the extracted sample to represent the entire dataset given a fixed confidence interval.

With a considerable manual effort, a tracking expert performed an exhaustive analysis of each sample both statically and dynamically, corroborating that 257 out of 273 scripts were actual web tracking (94.1%). These results are nearly the same as the ones obtained in the 10-fold cross validation described in §2 (94.4%).

Unknown Tracking in Blacklisted Domains Unknown tracking downloaded from blacklisted domains appeared in 70.49% of the websites with scripts (see Table 5). Only 7,455 blacklisted domains (see Table 6 for the 10 most prevalent

Table 7: Distribution of unknown tracking candidates in domain types.

Domain	# Samples	# Uniques
HTML	4,145,542	2,744,244
1st Party	679,319	283,337
3rd Party	796,158	241,578
TOTAL	*5,621,019*	*3,245,238*

Table 8: Popular clusters per domain type.

Domain	Cluster	% Scripts
HTML	Downloader	24.53%
	Statistics	13.47%
	Social Sharing	3.33%
1st Party	Statistics	26.70%
	Stateless Tracking	15.98%
	Advertisement	11.16%
3rd Party	Statistics	20.86%
	Stateless Tracking	15.27%
	Advertisement	14.08%

ones) hosted 3,923,615 of this type. This number is much smaller than in the case of known or completely unknown tracking scripts. In particular, script samples from `facebook.com` were present in 18.42% of the websites with scripts.

New Unknown Potential Tracking

Scripts & Domains Unknown tracking candidates represented 58.89% of the likely unknown tracking samples flagged by the *Unknown Tracking Analysis*. Their presence varied with regards to whether the domain was blacklisted or not. The prevalence of completely unknown tracking was higher than blacklisted ones: 90.69% of the websites with scripts used unknown tracking (see Table 9).

Due to the high number of discovered scripts, we performed an analysis to understand their nature. To this end, we measured the number of unique scripts and the domain type (third-party, first-party, and HTML-embedded) and performed a clustering analysis to find the most common behaviors.

We removed identical versions through code hashing and measured the number of unique scripts in each domain type (see Table 7). Usually tracking is used with third-party domains, but we found that is not the case for completely unknown tracking. Most of them (73.75%) were embedded in the HTML, bypassing current blacklisting solutions. 57.73% of the completely unknown were unique by hash. Only a small number of them repeated versions across different domain types (1st and 3rd party, usually), indicating that is not a common practice.

Table 9: Completely unknown tracking prevalence in websites.

# websites	831,677
– % in websites with tracking	92.95%
– % in websites (considering sites with scripts)	90.69%

Table 10: 10 most popular top-level domains hosting previously unknown tracking script candidates.

Domain	**# Websites**
`disquscdn.com`	15,185
`vk.me`	9,228
`baidustatic.com`	4,848
`kxcdn.com`	4,189
`adformdsp.net`	2,958
`jivosite.com`	2,829
`yandex.net`	2,739
`st-hatena.com`	2,399
`gtimg.cn`	2,384
`bitrix.info`	2,374

We computed the prevalence of domains hosting unknown tracking. The 10 most popular domains (for the list see Table 10) were different e.g., CDNs, search engines, social networks, or advertisement companies.

Clustering To understand the 3,245,238 unique tracking candidate scripts, we performed a cluster analysis. To overcome the high overhead of performing a cluster analysis directly on the large number of scripts, we conducted a clustering analysis in two steps. In the first step we clustered the 957 known trackers in our *Script Database* to find the different categories. We chose the *Affinity Propagation* clustering [26] due to its ability to automatically compute the cluster membership of an unknown sample and because it does not need to specify the number of clusters. In the second step, we computed the closest cluster for each of the different unknown tracking script candidates through the previously calculated clusters. The most popular category contained *downloader* scripts which were the 16.53% of the tracking candidates. The second most popular cluster included 12.85% of the scripts and was formed of *statistics* tracking scripts. We also measured the clusters with regards to their hosting (see Table 8).

4 Case Studies

Script Renaming Techniques Blacklisting techniques only blocked 43.80% of the known scripts versions TrackingInspector detected due to *script renaming*. Over 35% of the samples of each original scripts changed their name. While

Table 11: Potential unknown device fingerprinting prevalence.

Type	**# Scripts**	**# Websites**	**# Domains**
Font	25,502	24,873	704
Canvas	2,810	2,776	290
Shared	320	320	45
Total	*28,632*	*27,818*	*1,037*

samples of 200 of the original scripts did not present any change, versions from other 95 original scripts always changed their name. We analyzed three scripts: `piwik.js`, `evercookie.js`, and `dota.js`. 58.33% of `evercookie.js` samples, 91.13% `piwik.js` versions, and 99.66% of `dota.js` changed their name.

Among the script renaming techniques, we defined the following categories: (i) *related script renaming*, (ii) *random/neutral script renaming*, (iii) *functionality script renaming*, and (iv) *misleading script renaming. Related script renaming* changes the name to one directly or indirectly related to another service or website using the original script. For example, some versions changed their name to `chrysler.js` and `dodge.js`. *Random/neutral script renaming* replaces the name randomly, such as `penguin2.js` and `welcome.js`. *Functionality script renaming* modifies the name describing their goal, e.g., `fingerprint.js`, and `tracking.js`. Finally, *misleading script renaming* scripts change their names to well-known ones e.g., `jquery.alt.min.js` and `j.min.js`.

Canvas and Font Fingerprinting Canvas fingerprinting [39] and font probing [15] are two device fingerprinting techniques. Due to their relevance, we studied them to determine how many unknown scripts of this type our tool found.

We implemented two experiments for each type. These were designed to filter the potential tracking script samples that match rules for font probing and for canvas fingerprinting. We built two set of rules based on the scripts found by [3] and [2]. 710 unknown unique device fingerprinting scripts were found: 408 show canvas fingerprinting behavior, 247 font probing behavior, and 55 both.

We also used the new device fingerprinting scripts as the *Script Database* and performed a *Known Tracking Analysis* to measure their prevalence in the Alexa top 1M as well as the domains used for hosting them (see Table 11). 28,632 samples were detected and 27,818 websites used these unknown scripts.

We also analyzed the top potential device fingerprinting unknown script. The most prevalent unknown font probing script was `buttons.js`. It was hosted by *sharethis.com*, a well-known social widget for sharing content in social networks. The most used canvas fingerprinting unknown script was `Admeta.js`, downloaded by 981 websites from *atemda.com*, an ad provider. Regarding scripts containing both techniques, `image.js` was the most common unknown script, downloaded by 131 websites from the domain `magnuum.com`, a content delivery network.

Table 12: Comparison with Related Work since 2012.

Approach	**Type**	**Stateful**	**Stateless**	**Variations**	**Unknown**	**Size**
Blacklisting Solutions	Generic	✓	✓	✗	✗	–
Roesner et al., 2012 [43]	Specific	✓	✓	✗	✗	2,098
Cookieless Monster [41]	Specific	✓	✓	✗	✗	10,000
FPDetective [3]	Specific	✗	✓	✗	✓	1,000,000
The Web Never Forgets [2]	Specific	✗	✓	✗	✓	1,000,000
TrackingExcavator [32]	Specific	✓	✓	✗	✓	10,000[a]
Englehard et al, 2016 [17]	Specific	✓	✓	✗	✓	100,000/1,000,000[b]
TrackingInspector	Generic	✓	✓	✓	✓	1,000,000

[a] They analyzed 500 websites per year in the period 1996-2016.
[b] 100,000 stateful and 1,000,000 for stateless fingerprinting.

Fingerprinting-driven Malware Some reports [44, 51] linked targeted malware campaigns with fingerprinting. According to our results, sites with only tracking tend to be questionable and malicious. Hence, we inspected domains hosting only tracking and discovered suspicious scripts. For example, a script performed a fingerprinting step that included identification of the browser, *Java*, *Flash Player*, *SilverLight*, and the presence of a Chinese antivirus and then performed suspicious calls. It is important to remark, that the malicious script discovery was performed manually, based on the fingerprinting behavior and our findings, but we did not build any specific detection technique.

`101.99.68.18` and `202.172.54` hosted the script. `101.99.68.18` was allocated in the ISP *Piradious-NET* and `202.172.54` in *M1 Connect Pte. Ltd.*, known for hosting malware. The script was `jquery.min.js`, as in the library *jQuery* and was not obfuscated. Two Chinese sites in the Alexa top 1M (`52lif-e.cc` and `examres.com`) contained the iframe used to gather the malicious script.

By searching the iframe name, we found that this script was part of the *Chinad* botnet, as reported by *MalwareBytes* [35,36]. We did not perform a manual malware analysis, since it has already been performed by them. This is a exploit kit that compromised Chinese sites to fingerprint users looking for vulnerable components: e.g., in *Java* (CVE-2011-3544 and CVE-2012-4681), *Internet Explorer* (CVE-2014-6332), and *Flash* (CVE-2015-0311) and downloads versions of the *Chinad* botnet to perform DDoS attacks. To the best of our knowledge, the websites where we found the exploit kit were not previously reported.

5 Related Work

Previous work

Due to the concern that web tracking raised to users' privacy, a hectic research has been done to analyze these techniques.

One of the first tracking analyses that included HTML cookies was the one performed in [31]. Following this work, Mayer & Mitchell [37] studied different techniques for tracking including their policies and developed a tool to measure web privacy. Roesner et al. [43] presented a taxonomy for third-party tracking using cookies, measuring their presence. Nikiforakis et al. [41] studied three known fingerprinting companies and discovered that 40 websites within the Alexa top 10K sites used techniques such font probing. Acar et al. [3] discovered 404 sites in the top 1M using JavaScript-based fingerprinting and 145 sites within the Alexa top 10K sites using Flash-based fingerprinting. Also, Acar et al. [2] found a 5% prevalence of canvas fingerprint in the Alexa top 1M sites. They also found respawning by Flash cookies on 10 of the 200 most popular sites and 33 respawning more than 175 HTTP cookies. In the topic of web vulnerabilities, a previous analysis of the JavaScript included [40] determined the correlation between the trust of the included scripts as well as the domains hosting the scripts. However, our work differs from this large-scale study since we focus specifically in web tracking rather than vulnerabilities. Lerner et al. [32] presented *TrackingExcavator* and performed a retrospective analysis of tracking evolution since 1996, showing that it increased over time. Englehardt & Narayanan [17] analyzed the Alexa top 100K websites with regards to stateful tracking and the top 1M regarding stateless fingerprinting using blacklists and static rules, finding new device fingerprinting techniques. Our work differs in many ways (see Table 12 for a detailed comparison). We use the following parameters for the comparison:

- **Specific/Generic:** Specific approaches focus on concrete web tracking types using ad-hoc heuristics. A generic approach detects web tracking using a single approach for every type of tracking. Although some previous works deal with both stateful and stateless web tracking, we consider generic approaches the ones that do not rely on specific solutions or heuristics for each type.
- **Stateful/Stateless tracking:** We divide previous approaches into the ones devoted to detect classic stateful approaches, the ones that detect stateless ones, or the ones that detect both types.
- **Detection of Variants:** TrackingInspector detects variants of the known scripts. Other works, normally the ones based on heuristics, may also detect variants but they could not classify them as so.
- **Detection of Unknown:** TrackingInspector also detects potentially unknown tracking. Some heuristics-based methods are also able, but just when the script follows the defined heuristics.
- **Size:** To compare the soundness of our study, we measure the size in websites.

Summarizing, we believe that the major breakthroughs of our work are the following. First, TrackingInspector, a truly generic web tracking detector

```
//https://browserleaks.com/canvas
var ctx = canvas.getContext('2d');
var txt = "exampleText. <canvas> 1.0";
ctx.textBaseline = "top";
ctx.font = "14px 'Arial'";
ctx.textBaseline = "alphabetic";
ctx.fillStyle = "#f60";
ctx.fillRect(125,1,62,20);
ctx.fillStyle = "#069";
ctx.fillText(txt, 2, 15);
ctx.fillStyle = "rgba(102, 204, 0, 0.7)";
ctx.fillText(txt, 4, 17);

//Extracting image (detectable)
md5(ctx.getImageData(0, 0, canvas.width, canvas.height).
    data.toString())
md5(canvas.toDataURL())

//Extracting image (undetectable)
imageData = ""
for (i = 0; i < canvas.width/10; i++) {
  for (j = 0; j < canvas.height/10; j++) {
    imageData += ctx.getImageData(i*10, j*10, 10, 10).
        data.toString()
  }
}
md5(imageData)
```

Fig. 1: Snippet of OpenWPM detectable and undetectable canvas fingerprinting image extraction techniques.

that does not depend on blacklists or specific rules but on the previously known tracking scripts. Second, we discovered more than 3 million new unique tracking script candidates, a number higher than any reported previous work. We also found 710 new potential device fingerprinting scripts: 408 canvas fingerprinting, 247 font probing, and also 55 that exhibited both fingerprinting behaviors, a number also higher than previous work. Our tool also automatically detected the fingerprinting phase of a targeted malware campaign for the first time to date.

Empirical comparison

We also implemented a comparative experiment that consist of 2 specific case studies, showing how our tool can detect instances of tracking scripts that are currently not flagged by other tools.

Canvas Fingerprinting. Englehard et al. [17] analyzed Alexa top 1M websites, using OpenWPM [18], looking for tracking scripts (canvas fingerprinting included). For this specific technique, they proposed a set of 4 rules as a filtering criteria. Even if theses rules could definitely help in the detection process, there is an extremely simple evasion that any script could perform in order to completely avoid detection. The exact rule that allows it, is the following: *"The script extracts an image with `toDataURL` or with a single call to `getImageData` that specifies an area with a minimum size of 16px x 16px."* If the script performs multiple calls to `getImageData` without exceeding the size determined in the rule, the analysis will erroneously flag the script as non-tracking (see Figure 1). In order to check if our approach would detect this case, we prepared two different scenarios.

1. We modified a script known for performing canvas fingerprinting (`dota.js`) to obtain the data following the aforementioned evasion technique. Then, we performed a *Known Tracking Analysis* to see if this change would be enough to evade the detection of the script as a variation of the original script. TrackingInspector was able to identify the script with a similarity of 99% and flags it as a modification of a known tracking script present in the *Script Database*.
2. In this case, instead of using a known script as a baseline, we created a new tracking script from scratch, but based on the same idea. As in the previous scenario, we implemented the data retrieval process following the evasion technique described. We performed an *Unknown Tracking Analysis* in this script to verify the effectiveness of the tool in this exact case. TrackingInspector was able to correctly classify the script as an unknown tracking script.

In conclusion, in both of the presented scenarios, our tool was able to detect the tracking script using its different components, in one case to detect the variation and in the other to detect the fingerprinting technique; while the previous approaches failed.

Font Fingerprinting. This type of fingerprinting technique has been extensively analyzed using both OpenWPM [17] and FPdetective [3]. One of the criteria used in these works to classify the script as a candidate of performing this kind of tracking, is the number of different fonts loaded (50 and 30, respectively). However, the work of Fifield and Egelman [20] allows to generate a specific type of font fingerprinting that measures the bounding boxes of some specific Unicode code points. For this case, the technique just needs five generic families (e.g., sans-serif, serif, monospace, cursive, and fantasy). As the number of different fonts used is much lower than the ones specified in their respective rules, neither of the previously mentioned analysis were able to detect them. In contrast to the previous case, this is not a small modification of the technique to avoid detection, but another font fingerprinting approach.

Therefore, we have created a fingerprinting script following this font fingerprinting technique from scratch. Then, we performed an *Unknown Tracking*

Analysis to verify if our tool was able to detect this script as a tracking script. TrackingInspector correctly classified this script in the tracking group. As our approach is not based in strict human-generated rules, but in a learning progress with known scripts performing various types of tracking techniques, it can detect different fingerprinting attempts without explicit rules.

6 Conclusions

TrackingInspector measured the web tracking prevalence in websites and domains. The results show that web tracking is very extended and that current solutions cannot detect every known or unknown tracking. We also examined the hiding techniques used to avoid blacklists, finding different script renaming techniques. TrackingInspector detected both known or variations of tracking and likely unknown web tracking. We also found new potential stateless device fingerprinting and their prevalence, showing that even well-known companies provide them. Among the discovered unknown web tracking, we found a previously reported malware campaign targeting Chinese websites, showing that malicious activities may exhibit fingerprinting behavior. We believe that *fingerprinting-driven malware* may become a relevant issue in the future.

Acknowledgments

We would like to thank the reviewers for their insightful comments and our shepherd Nick Nikiforakis for his assistance to improve the quality of this paper. This work is partially supported by the Basque Government under a pre-doctoral grant given to Iskander Sanchez-Rola.

References

1. Abine: Tracking list. `https://www.abine.com/index.html` (October 2017)
2. Acar, G., Eubank, C., Englehardt, S., Juarez, M., Narayanan, A., Diaz, C.: The web never forgets: Persistent tracking mechanisms in the wild. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS) (2014)
3. Acar, G., Juarez, M., Nikiforakis, N., Diaz, C., Gürses, S., Piessens, F., Preneel, B.: FPDetective: dusting the web for fingerprinters. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS) (2013)
4. AdblockPlus: Easyprivacy. `https://easylist.adblockplus.org/` (October 2017)
5. AdGuard: Tracking list. `https://adguard.com/` (October 2017)
6. Blocksi: Web content filtering. `http://www.blocksi.net/` (October 2017)
7. Canali, D., Cova, M., Vigna, G., Kruegel, C.: Prophiler: a fast filter for the large-scale detection of malicious web pages. In: Proceedings of the 20th international conference on World Wide Web (2011)
8. Canali, D., Lanzi, A., Balzarotti, D., Kruegel, C., Christodorescu, M., Kirda, E.: A quantitative study of accuracy in system call-based malware detection. In: Proceedings of the International Symposium on Software Testing and Analysis (2012)

9. Cliqz: Ghostery. `https://www.ghostery.com/` (October 2017)
10. Cloudacl: Web security service. `http://www.cloudacl.com/` (October 2017)
11. Cova, M., Kruegel, C., Vigna, G.: Detection and analysis of drive-by-download attacks and malicious javascript code. In: Proceedings of the 19th international conference on World Wide Web (2010)
12. Curtsinger, C., Livshits, B., Zorn, B.G., Seifert, C.: ZOZZLE: Fast and Precise In-Browser JavaScript Malware Detection. In: Proceedings of the USENIX Security Symposium (Sec) (2011)
13. Data, N.: Search engine for source code. `https://nerdydata.com/` (October 2017)
14. Disconnect: Tracking list. `https://disconnect.me/` (October 2017)
15. Eckersley, P.: How unique is your web browser? In: Proceedings of the Privacy Enhancing Technologies (PETS). Springer (2010)
16. van Eijk, R.: Tracking detection system (tds). `https://github.com/rvaneijk/ruleset-for-AdBlock` (October 2017)
17. Englehardt, S., Narayanan, A.: Online tracking: A 1-million-site measurement and analysis. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS) (2016)
18. Englehardt, S., Narayanan, A.: Openwpm. `https://github.com/citp/OpenWPM` (October 2017)
19. Fanboy: Tracking list. `https://www.fanboy.co.nz/` (October 2017)
20. Fifield, D., Egelman, S.: Fingerprinting web users through font metrics. In: International Conference on Financial Cryptography and Data Security. pp. 107–124. Springer (2015)
21. FileWatcher: The file search engine. `http://www.filewatcher.com/` (October 2017)
22. Fisher, R.A.: Statistical methods and scientific inference. Hafner Publishing Co. (1956)
23. Fortinet: Fortiguard web filtering. `http://www.fortiguard.com/` (October 2017)
24. Foundation, E.F.: Privacy badger. `https://www.eff.org/es/privacybadger` (October 2017)
25. Foundation, M.: Public suffix list. `https://publicsuffix.org/list/` (October 2017)
26. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. Science 315(5814), 972–976 (2007)
27. Hidayat, A.: Phantomjs. `http://phantomjs.org/` (October 2017)
28. Ho, T.K.: Random decision forests. In: Proceedings of the International Conference on Document Analysis and Recognition (ICDAR) (1995)
29. Kamkar, S.: Evercookie. `https://github.com/samyk/evercookie` (October 2017)
30. Kaspersky: Tracking list(abbl). `http://forum.kaspersky.com/` (October 2017)
31. Krishnamurthy, B., Wills, C.: Privacy diffusion on the web: a longitudinal perspective. In: Proceedings of the International Conference on World Wide Web (WWW) (2009)
32. Lerner, A., Simpson, A.K., Kohno, T., Roesner, F.: Internet Jones and the Raiders of the Lost Trackers: An Archaeological Study of Web Tracking from 1996 to 2016. In: Proceedings of the USENIX Security Symposium (SEC) (2016)
33. Lielmanis, E.: Js beautifier. `http://jsbeautifier.org/` (October 2017)
34. Luhn, H.P.: A statistical approach to mechanized encoding and searching of literary information. IBM Journal of Research and Development 1(4), 309–317 (1957)
35. MalwareBytes: Unusual exploit kit targets chinese users (part 1). `https://blog.malwarebytes.org/exploits-2/2015/05/unusual-exploit-kit-targets-chinese-users-part-1/` (accessed October 2017)

36. MalwareBytes: Unusual exploit kit targets chinese users (part 2). https://blog.malwarebytes.org/intelligence/2015/06/unusual-exploit-kit-targets-chinese-users-part-2/ (accessed October 2017)
37. Mayer, J.R., Mitchell, J.C.: Third-party web tracking: Policy and technology. In: Proceedings of the International Symposium on Security and Privacy (Oakland) (2012)
38. MeanPath: The source code search engine. https://meanpath.com/ (October 2017)
39. Mowery, K., Shacham, H.: Pixel perfect: Fingerprinting canvas in HTML5. Proceedings of the Web 2.0 Workshop on Security and Privacy (W2SP) (2012)
40. Nikiforakis, N., Invernizzi, L., Kapravelos, A., Van Acker, S., Joosen, W., Kruegel, C., Piessens, F., Vigna, G.: You are what you include: large-scale evaluation of remote javascript inclusions. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)
41. Nikiforakis, N., Kapravelos, A., Joosen, W., Kruegel, C., Piessens, F., Vigna, G.: Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In: Proceedings of IEEE Symposium on Security and Privacy (Oakland) (2013)
42. Rieck, K., Krueger, T., Dewald, A.: Cujo: efficient detection and prevention of drive-by-download attacks. In: Proceedings of the Annual Computer Security Applications Conference (CSS) (2010)
43. Roesner, F., Kohno, T., Wetherall, D.: Detecting and defending against third-party tracking on the web. In: Proceedings of the USENIX conference on Networked Systems Design and Implementation (NDSI) (2012)
44. Security Response, Symantec: The Waterbug attack group. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/waterbug-attack-group.pdf (2015)
45. Selzer, A.: Beaverbird. https://github.com/AlexanderSelzer/BeaverBird (October 2017)
46. Services, A.W.: Alexa top sites. https://aws.amazon.com/es/alexa-top-sites/ (October 2017)
47. Shekyan, S., Vinegar, B., Zhang, B.: Phantomjs hide and seek. https://github.com/ikarienator/phantomjs_hide_and_seek (October 2017)
48. Singer, N.: Do not track? advertisers say "don't tread on us". http://www.nytimes.com/2012/10/14/technology/do-not-track-movement-is-drawing-advertisers-fire.html (2012)
49. Sparck Jones, K.: A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation 28(1), 11–21 (1972)
50. Technologies, A.: Privacyfix tracking list. http://privacyfix.com (October 2017)
51. Threat Intelligence, FireEye: Pinpointing Targets: Exploiting Web Analytics to Ensnare Victims. https://www2.fireeye.com/rs/848-DID-242/images/rpt-witchcoven.pdf (2015)
52. of Trust, W.: Crowdsourced web safety. https://www.mywot.com/ (October 2017)
53. TrustArc: Truste tracking list. https://www.truste.com/ (October 2017)
54. Vasilyev, V.: Fingerprintjs. https://github.com/Valve/fingerprintjs (October 2017)
55. Webutation: Open website reputation. http://www.webutation.net/ (October 2017)
56. Yamaguchi, F., Lindner, F., Rieck, K.: Vulnerability extrapolation: assisted discovery of vulnerabilities using machine learning. In: Proceedings of the USENIX Conference on Offensive Technologies (WOOT) (2011)