Known and Unknown Generic Web Tracking Analyzer: A 1 Million Websites Study

Iskander Sanchez-Rola and Igor Santos
DeustoTech , University of Deusto
`{iskander.sanchez, isantos}@deusto.es`

Technical Report

Abstract

Web tracking is a widespread technique on the Internet to gather user data. While tracking may not pursue user damage, it may violate user consent and privacy. In addition, some recent reports have linked web tracking with targeted malware campaigns. Recent research studied well-known and advanced tracking techniques. Despite the fact that these works improved the understanding of the current tracking landscape, they were not intended to generically detect and understand all types of web tracking techniques. In this paper, we present the first general large-scale analysis of different known and unknown web tracking scripts on the Internet to understand its current ecosystem and behavior. To this end, we implemented TRACKINGINSPECTOR the first automatic method to detect generic web tracking scripts. This technique automatically retrieves scripts of a website and, through code similarity and machine learning, detects modifications of known tracking scripts and discovers unknown web tracking script candidates. TRACKINGINSPECTOR analyzed the Alexa top visited 1,000,000 websites, computing the 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 discovered and we examined the script renaming techniques they used to avoid blacklists. In addition, 5,500,000 completely unknown likely tracking scripts were found, including more than 700 new different potential device fingerprinting (canvas and font probing) 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.

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. Many methods exist for web tracking: *cookies* [1] that utilize user's machine to store information, *device fingerprinting* that computes a unique id for each machine (e.g., by retrieving the installed components such as fonts [2], the computed difference in rendering text by the HTML5 Canvas API [3], or the combination of attributes of different browsers [4]), *evercookies* that bypass user cookie cleaning by exploiting browser storage, *cookie syncing* that allows trackers to share users' fingerprints, and *ETags* inserted within images used to check the user identity [5]. Some of these techniques (e.g., cookies or HTML5 storage) require a stateful user navigation to store user environment and variables. In contrast, stateless techniques do not require storage to identify a user and, therefore, cannot be blocked by common private browsing configurations.

Despite the fact that web tracking poses a threat to users' privacy and anonymity, it is not usually considered harmful by itself. Indeed, web advertising companies defend web tracking as a fundamental component of the web economy [6]. However, recent reports by *Symantec* [7] and *FireEye* [8] discovered two different targeted malware campaigns that employed a fingerprinting stage to increase the exploit success rate and to hide their exposure.

Recent work studied both stateful [9–11] and stateless [12, 13] web tracking, raising a general concern in the community about the prevalence of these techniques on 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. An unpublished work [14], parallel to our paper, 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 and discovery cannot be performed with these methods.

Given this background, we present here the first large-scale analysis of generic web tracking scripts. Due to the limitations of current tracking analysis solutions, we build our own tracking inspection 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[1] top 1,000,000 websites in order 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 blacklistings solutions limit or block most web tracking on the Internet?, and (iv) can TRACKINGINSPECTOR discover new web tracking scripts? To what extent?

[1]`http://www.alexa.com/`

The major contributions and findings of this paper are:

- The first global large-scale study of generic web tracking. Our results show that more than 90% of the websites performed tracking and that more than 50% of the scripts exhibited tracking behavior. We also computed the most prevalent domains hosting tracking and analyzed their ecosystem.
- TRACKINGINSPECTOR, the very first tool to automatically detect generic web tracking scripts through code similarity and machine learning. The results of the large-scale study show its ability to automatically detect known tracking scripts and their modifications, and to discover potentially unknown web-tracking scripts. Our method was able to detect more than 2,000,000 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 anti-tracking solutions. Hence, we studied these hiding techniques, discovering several *script renaming techniques* that script versions are using.
- More than 5,500,000 new scripts that exhibited tracking behavior not previously reported or blacklisted. 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.

The remainder of this paper is organized as follows. Section 2 presents TRACKINGINSPECTOR, our automatic web tracking analyzer. Section 3 describes the large-scale analysis and outlines the major global findings of this study. Section 4 describes selected interesting case studies extracted from our results. Section 5 discusses the main findings of this work and their implications. Section 6 contextualizes this work with recent related work. Finally, we extract our conclusions in Section 7.

2 Tracking Analysis & Detection

2.1 General Description

Current Solutions

To understand what is the current landscape of tracking scripts on the Internet, we evaluated the available solutions for tracking detection: blacklisting (e.g., *EasyPrivacy*[2] and *Ghostery*[3]), the EFF's tool *Privacy Badger*[4] based on simple heuristics, *FPDetective* framework [12], and *OpenWPM* solution.[5]

Blacklisting tools rely on a list with 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 of tracking behavior but named with the same script name as one within the blacklist.

[2]https://easylist.adblockplus.org/
[3]https://www.ghostery.com/
[4]https://chrome.google.com/webstore/detail/privacy-badger/
[5]https://github.com/citp/OpenWPM

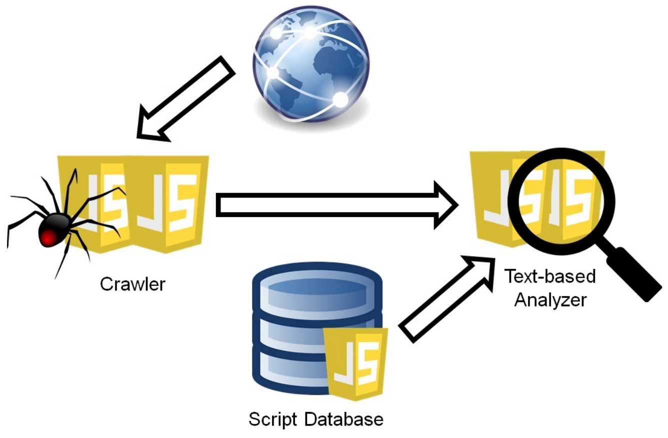

Figure 1: General overview of TRACKINGINSPECTOR.

The heuristics used in the *Privacy Badger* plugin block third-party cookies. This approach raises false positives and only focuses in cookies. Nevertheless, web tracking adopts many other different forms that are not covered by this tool.

FPDetective and *OpenWPM* are tracking analysis frameworks based on blacklisting and/or rules to detect several tracking behaviors. These frameworks solve the main limitations of sole blacklisting techniques. However, as these techniques are based on predefined rules, the tracking script can be modified with techniques that bypass the existing rules. Although new rules can be included, the process requires a considerable manual effort and rules are fixed for the detection of specific fingerprinting techniques, while our goal is a generic automatic tracking analysis.

Due to the limitations of current solutions for our goal, we decided to implement TRACKINGINSPECTOR, our own tracking detection method to answer our research questions.

TRACKINGINSPECTOR Solution

TRACKINGINSPECTOR (see Figure 1) is composed of three main components: (i) a *Crawler*; (ii) a *Script Database* with both tracking and non-tracking scripts, and (iii) a *Text-based Analyzer*.

The basic functionality of TRACKINGINSPECTOR starts by downloading scripts through the *Crawler*. To this end, we implemented a set of heuristics to maximize the number of scripts downloaded by the *Crawler* from each website, improving the methods previously proposed in the literature. The *Crawler* waits until every script in the site is downloaded; including third-party, first-party, or HTML-embedded scripts.

These scripts are then analyzed by the *Text-based Analyzer* through its two analysis components:

1. The *Known Tracking Analysis* measures the similarity of each script with the tracking and non-tracking scripts stored in the *Script Database*. The script under inspection can be detected as a variation of a particular known

tracking script based on their code similarity or it can be flagged as a non-tracking script. 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 tracking scripts not previously known or variations of known scripts whose differences are enough to consider them unknown. This component is devoted to discover new tracking scripts to be added to the *Script Database* for the *Known Tracking Analysis*.

In contrast with other methods devoted to detect specific tracking [12], TRACKINGINSPECTOR does not perform a complete dynamic analysis of the scripts. Instead, the *Crawler* dynamically retrieves all the scripts of the website and, then, the script is analyzed by the *Text-based Analyzer*. To detect and block tracking, dynamic tracking detection approaches monitor the website behavior during browsing and compare it with specific rules set for concrete types of web tracking. Otherwise, TRACKINGINSPECTOR is oriented to detect any sort of web tracking and not only specific types of it and, therefore, it requires the general and complete behavior of a website under inspection. In order to adapt a dynamic approaches to our desired generic tracking detection, we need to construct a rule database with every possible set of rules to every type of known tracking behavior. However, such an approach will lack TRACKINGINSPECTOR'S capabilities to generically detect all variations of known tracking behaviors and will not be capable to discover unknown tracking candidates.

2.2 Crawler

Overview

The *Crawler* component is devoted to automatically retrieve every JavaScript file statically or dynamically loaded in a website. The *Crawler* is based on *PhantomJS*,[6] a well-known headless browser. To avoid the typical detection of the automated browsing when using headless browsers, we adapted existing advanced hiding methods[7] to disguise the browser as a common one. With this adaptation, our *Crawler* is capable of performing a 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 website 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, rather than waiting a fixed time and gathering them, like previous work on fingerprinting detection [12, 13], we empirically analyzed the different script generation behaviors in 250,000 random websites within the Alexa top 1M websites to determine a global methodology for web analysis and to 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. When all frames are loaded before the 60 seconds are elapsed, the *Crawler* waits for other additional 15 seconds (or until the maximum time is elapsed). 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 since additional scripts were triggered in that time. The computation time takes into account the possible redirections within the website or its frames. Then, the *Crawler* retrieves the resulting HTML code, with all the generated possible modifications, and gathers the scripts embedded in the HTML code.

Once the scripts have been gathered, the *Crawler* starts a deobfuscation phase and finally, the *Crawler* starts a process of cleaning duplicate scripts, files that are not actually JavaScript, or empty files.

De-Obfuscation

We implemented a deobfuscator based on *JSBeautifier*,[8] a well-known online deobfuscator. Using the techniques implemented in this tool, our method tries to unravel the original code, checking in each iteration whether or not the script has been completely deobfuscated with the metrics specified for each obfuscator.

In this way, we can deal with multiple layers and multiple known techniques of obfuscation. In fact, we discovered some JavaScript obfuscators that use other obfuscators iteratively to perform a multiple layer obfuscation. When the script is deobfuscated, our method performs an additional step to deal with one obfuscator with a particular behavior.

This obfuscator, 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 of this technique that the script has been executed. Our deobfuscator retrieves the code stored in the string and unescapes it, discarding the additional code within the script.

2.3 Script Database

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

In order to generate the tracking scripts dataset, we retrieved scripts on blacklists, open-source projects, academic papers, and also tracking scripts found during the manual inspection when searching for non-tracking scripts:

- **Blacklists:** We used *EasyPrivacy*, *Kaspersky Tracking List (ABBL)*,[9] *ABINE*,[10] the tracking version of *AdGuard*,[11] the

[6] `http://phantomjs.org/`

[7] `https://github.com/ikarienator/phantomjs_hide_and_seek`

[8] `http://jsbeautifier.org/`

[9] `http://forum.kaspersky.com/`

[10] `https://www.abine.com/index.html`

[11] `https://adguard.com/`

tracking list *FanBoy*,[12] and the *Tracking Detection System (TDS)* by Rob van Eijk.[13] These lists were selected because they include scripts and not just domains. We omitted blacklisted domains in these lists 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 Section 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 included this type of scripts.

- **Open-source Tracking Projects:** We retrieved several open-source tracking projects to complement the information already stored in the blacklisting services: *BeaverBird*,[14] *FingerPrintJS*,[15] and *Evercookie*.[16]

- **Academic Papers:** Regarding academic papers, we also included in our tracking *Script Database* the scripts found in [12,13]. With these scripts, we complement the information of the *Script Database* with newly advanced types of tracking scripts.

We processed the list of potential scripts and stored the scripts whose complete URL was available. However, some scripts were not available. In those cases, we tried to download them through `archive.org`, removing all the service-related text and code afterwards. In other cases script names were only available and we searched it using several code searchers (e.g., *meanpath*,[17] *NerdyData*,[18] *FileWatcher*[19]), or common search engines (e.g., *Google* and *Bing*). We manually checked each script afterwards to determine whether it was actually tracking or not.

Since some of the scripts were obfuscated, we applied the deobfuscation process already described in section 2.2. Afterwards, we performed an additional step to remove duplicates or different versions of scripts. To this end, we modeled the code of each script using a *Bag of Words* (BOW) in a *Vector Space Model* (VSM) approach [15], computing the cosine similarity of each script within the *Script Database*. After an empirical testing of different versions of various scripts, we defined 0.85 as the similarity threshold for a script to be considered as a version or modification of another, adding the original scripts to *Script Database*. However, we also added a small number of script versions that, albeit being considered modifications of original scripts, they presented new functionalities or behaviors that TRACKINGINSPECTOR should be capable of detecting. After this process, 957 original tracking scripts were stored in our *Script Database*. To balance the *Script Database*, we randomly selected 957 non-tracking scripts from the aforementioned sources. We also added some additional non-tracking scripts that may be incorrectly flagged as web tracking (e.g., `jquery.cookies.js`, `jquery.form.min.js`, or `jquery-ui.js`) in order to avoid false positives in the *Unknown Tracking Analysis* phase. These scripts will only be used in the *Known Tracking Analysis* phase and not in the *Unknown Tracking Analysis* since training may be biased, losing accuracy.

[12]`https://www.fanboy.co.nz/`
[13]`https://github.com/rvaneijk/ruleset-for-AdBlock`
[14]`https://github.com/AlexanderSelzer/BeaverBird`
[15]`https://github.com/Valve/fingerprintjs`
[16]`https://github.com/samyk/evercookie`
[17]`https://meanpath.com/`
[18]`http://nerdydata.com/`
[19]`http://www.filewatcher.com/`

2.4 Text-based Analyzer

Overview

The *Text-based Analyzer* is the component 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, modifications or variants of known scripts stored in the *Script Database* and a (ii) *Unknown Tracking Analysis*, whose goal is to automatically identify tracking script candidates.

In previous studies, tracking and fingerprinting behaviors were detected by means of blacklisting or manually-generated rules [12,13]. Since our goal is to perform a large-scale analysis of current tracking behaviors, we consider that these approaches were either limited in the case of blacklisting, or not feasible in the case of generating rules because our goal is not to analyze a particular tracking behavior but the global overview of web tracking.

Scripts Representation

We use the text representation of the script source code. There are different approaches to represent it.

In our case, 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, BOW approach captures the token frequencies to model the script. Despite the fact that ASTs have been widely applied for static analysis, in our preliminary tests, the text-categorization approach BOW behaved better to detect generic tracking behaviors, because ASTs model code syntax strictly taking also into account the script structure, BOW models the usage of tokens. Therefore, ASTs are worse when dealing with script modifications or new scripts than the standard BOW model. Since our goal is not to capture any particular behavior but to generically detect any type of web tracking, other approaches may overfit and fail to detect modifications or new web tracking scripts.

This approach has also its shortcomings. In particular, the BOW approach will take into account the tokens within the code including variables and identifiers. Nevertheless, since it is designed to capture all of them, a single modification of a variable name will never avoid detection. On the other hand, our approach may fail if someone willingly changes the identifiers to those typical on web tracking. However, there is no explainable reason why a legitimate script would perform such transformation, since the script will impersonate a web tracking script and will be blocked.

Text Representation of the Script Database

Although the goals of known and unknown approaches are different, they both represent scripts using the aforementioned BOW approach. As in the *Script Database*, all the scripts are modeled through a VSM, composed of the vocabulary of terms within the scripts. *Text-based Analyzer* represents each script as a sequence of each term frequencies, using the well-known *Term Frequency [16] – Inverse Document Frequency* (TF–IDF) [17] weighting schema. *Known Tracking Analysis* is dedicated to detect versions or modifications of currently known tracking scripts stored in the *Script Database*. To this end, this component computes the similarity of the script under inspection with known tracking scripts. When the already empirically computed threshold of 0.85 is surpassed, the script is flagged as a known tracking script version. Otherwise, *Known Tracking Analysis* compares the script under inspection with non-tracking scripts in the *Script Database* to avoid false positives in the *Unknown Tracking Analysis* phase. If the 0.85 threshold is surpassed, the script will be marked as non tracking.

Unknown Tracking Analysis is based on supervised machine learning. Using the aforementioned *Script Database*, we performed a 10-fold stratified cross-validation experimental evaluation with several well-known machine-learning algorithms to decide which classifier to use for this task. The best performing classifier was *Random Forest* [18] configured with 950 *Random Trees*. This classifier is an ensemble learning method for classification that generates several decisions trees at training and decides the classification based on the mode or mean of their partial classifications. In the training phase for Random Forest, the bagging technique is used to generate the weak random tree learners. Then, to build the aggregate, a similar but more general method is used to select the different high-level splits sometimes known as feature bagging.

Evaluation

The *Known Tracking Analyzer* using a 0.85 threshold did no report any false positives in our tests, and, as it is obvious, will be capable of detecting any modification of known tracking scripts that are, at least, 85% similar than the ones in the *Script Database*. Therefore, we believe that this component should be used in an end-user environment as a replacement of blacklisting techniques. Therefore, we will compare its performance with blacklisting solutions in Section 3.

We consider any script not detected by the *Unknown Tracking Analyzer* as *previously unknown*. These scripts, can only be detected by the extitUnknown Tracking Analyzer and not by any other method in the literature. During the cross-validation evaluation, this component achieved an area under the ROC curve of 0.982, a 93.6% accuracy, a true positive rate of 94.4%, and a false positive ratio of 7.5%. With this classifier, we built the *Unknown Tracking Analysis* to discover likely candidates of unknown tracking scripts. It is important to remark that this component is the first one in the literature capable to detect previously unseen generic tracking scripts and that it is devoted to the discovery of new tracking scripts rather than and end-user environment. Therefore, we believe that for this usage the reported false positive ratio is more than acceptable. However, in order to further evaluate the tracking discovery capabilities of the *Unknown Tracking Analysis* component, an additional evaluation of the method will be performed through a manual inspection of web tracking scripts in the wild in Section 3.

3 Large-Scale Analysis

3.1 Preliminaries

The main goal of this analysis is to answer the following research questions:

- *How widespread is web tracking on the Internet?*
- *What is the current ecosystem in web tracking provision and deployment on the Internet?*
- *Can current blacklisting solutions limit or block most web tracking on the Internet?*
- *Can* TRACKINGINSPECTOR *discover new web tracking scripts? To what extent?*

To this end, we selected the Alexa top 1M websites. The *Crawler* retrieved the scripts within the 1,000,000 websites and the *Text-based Analyzer* inspected them. When downloading scripts from a website that had been removed, the site was searched through `archive.org`. Since `archive.org` adds code and also files to the downloaded website, we performed a cleaning process. First, we removed any reference to `archive.org` from the domain. Second, we removed the scripts related with this service both external and embedded in the HTML. Finally, we removed any reference in the code comments to `archive.org`.

We failed to retrieve the scripts of 3.67% of the websites because its access was restricted (401 and 403), the site was not accessible, or it was not possible to find in `archive.org`.

Since TRACKINGINSPECTOR only uses the scripts of well-known blacklisting services in its *Script Database* but not the domains, some of the scripts flagged as unknown likely tracking scripts by the *Unknown Tracking Analyzer* may be only unknown by TRACKINGINSPECTOR but known by existing 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)*, *Disconnect*,[20] *Truste*,[21] *Ghostery*,[22] *Privacy Choice*,[23] *Privacy Badger*,[24] and *Web of Trust*[25] (the domains classified there as the category 301 - online tracking).

In order to retrieve the information about the web tracking ecosystem, we gathered several additional information about the hosted website and the top-level domains from where the scripts were downloaded. We also analyzed the domains, because, in other related topic, the hosted scripts in domains have proven to have a strong influence in the trust of the websites [19]. To correctly retrieve the top-level domains and to reduce the domain name to its origin, we used the `effective_tld_names.dat` by Mozilla.[26] Next, we extracted the country where the domains were hosted, which ISP gave access to the domain, the associated ASs, and the category of the website. To determine the category of the website,

[20]`https://disconnect.me/`
[21]`https://www.truste.com/`
[22]`https://www.ghostery.com/`
[23]`https://www.privacyfix.com`
[24]`https://www.eff.org/es/node/73969`
[25]`https://www.mywot.com/`
[26]`https://publicsuffix.org/list/`

we utilized three different services: *Cloudacl*,[27] *Blocksi*,[28] and *Fortiguard*.[29] The categories of these three services are similar, and, after a process of category name normalization, 78 category names were established. We also performed an analysis of the online reputation of the websites through the *Webutation* service,[30] that uses a combination of various users' feedback, comments, and also different analyses of the website such as *Google Safe Browsing, Norton Antivirus*, `phistank.com`, among others.

For the sake of clearness, we define and clarify the terminology that we will be using in the rest of the section to name and classify the web tracking and non-tracking scripts. Regarding the type of tracking, we can distinguish between:

- **Known Tracking Scripts:** We call known tracking scripts to script that are in the range between 85% and 100% similarity to the ones in the database.
- **Unknown Tracking Scripts:** We name this way to scripts that are not at least 85% different to the ones in the script database. In this way, these script can be either new scripts from scratch or versions of known scripts that have been modified enough to be considered *new* or *unknown*. These scripts are flagged by the unknown text-analyzer that has a small percentage of error and therefore we will also name them sometimes *tracking script candidates* or *likely tracking scripts*. In this category, we also distinguish between two sub-categories.
  - *Unknown Blacklisted Tracking Scripts:* These scripts are *unknown tracking scripts* whose hosting domain is in one of domain blacklisting domains but the script it is not. Therefore, the domain is known to be hosting web tracking but the script has not particularly flagged by any blacklist as web tracking.
  - *Completely Unknown Tracking Scripts:* These scripts are unknown and their hosting domains has not been blacklisted either.

Since there can be different samples or version of the same original unique script, we have also classified these three possibilities:

- **Unique Script:** Different scripts compared by hash. Versions of known tracking scripts that are not exactly equal are considered different unique scripts.
- **Original Script:** The unique source scripts in the script database.
- **Script Version or Script Sample:** Each occurrence or download of a script.

3.2 Tracking Ecosystem

General Overview

A total of 20,969,926 script samples (tracking and non-tracking scripts) were downloaded from the the Alexa top 1M websites. Impressively, nearly 60% of them were flagged as web tracking (see Table 1). In other words, more than the half of the functionality in the web is potentially devoted to track users. We also measured if this overwhelming number of tracking behaviour was hidden by obfuscation techniques. It is important to remark that we only consider obfuscation and not mimification or other techniques in this measurement, as they did in other studies [20] because our target is different. Although web tracking include lots of well-known and accepted techniques as analytics, we believed than more advanced and controversial techniques will try to hide their nature somehow. However, as opposite to initially expected, the number of scripts obfuscated is irrelevant: just a 0.45% of them. Likewise, only a 4.75% were downloaded via HTTPS, indicating lack of security in their communications.

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*

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*

There was also an interesting 46,277 of the websites did not use any scripts at all. However, after analyzing them, we found out that the size of the main page of these websites was very small (an average of 7.53KB), implying that these websites were very basic or not completely functional. Therefore, we computed the tracking prevalence percentage both in every analyzed website and also in websites with scripts (see Table 2), discovering that nearly every website performed web tracking.

Regarding the known tracking scripts, they represented only around a 20% of the tracking script samples, whereas the unknown samples represented an stunning majority. In this way, the majority of the flagged samples were not previously stored

[27]`http://www.cloudacl.com/`
[28]`http://www.blocksi.net/`
[29]`http://www.fortiguard.com/`
[30]`http://www.webutation.net/`

Table 4: Relation between *webutation* (by the reputation online service) and websites using some or only tracking scripts. *% Some T.* means the percentage of websites with some tracking scripts and *% Only T.* means the percentage of websites with only tracking scripts.

Category	**# Websites**	**% Some T.**	**% Only T.**
Red	6,322	96.98%	15.41%
Yellow	11,329	97.48%	15.31%
Grey	37,3050	96.74%	6.45%
Green	518,069	98.21%	5.95%

in the *Script Database*. However, using the previously omitted domain blacklists, we discovered that 41.11% of these unknown scripts were in blacklisted domains (see Table 3 for details), being unknown blacklisted scripts. Anyhow, the number of scripts that neither the script and the hosting domain were blacklisted, was the impressive majority of 46.90%.

The web tracking scripts were downloaded from a total of 891,873 different top-level domains. In particular, known tracking script samples were downloaded from 540,369 different top-level domains, while unknown tracking scripts were hosted in 841,425 different domains.

To understand the demographics of tracking usage and provision, we analyzed several features of the websites that downloaded the script and, also, of the domains that are providing these scripts.

Website Demographics

To obtain a better understanding of which websites performed more tracking, we analyzed their different aspects and their tracking prevalence, omitting, for obvious reasons, websites without scripts. To discover the relevance of web tracking in each analyzed aspect (website category and webutation in both cases, and origin country and network entities in the case of domains), we run several preliminary tests computing the differences of several tracking script ratios per website to determine which ratio was able to discriminate between the examined categories. The results showed that computing the number of websites with only tracking scripts per each studied aspect eased the discrimination, while the number of websites with some tracking and the number of scripts per aspect showed the overall behavior.

We computed the following ratios: (i) percentage of tracking scripts per script in the category, (ii) percentage 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 script percentage were, among others, *personal websites*, *hacking*, *spyware and adware*, *social networks*, or *peer to peer* websites. On average, each category included 97.78% ± 1.45 of websites with some tracking. Seven of the categories included 100% tracking websites. However, these concrete categories had a low number of websites ranging from just 1 to 79. The average percentage of websites per category that only contained tracking scripts was much lower: 7.50% ± 4.53. In this case, there was an important difference between categories: the highest ratio of only tracking websites included was 26.67% while the lowest was 0.00%. The top categories were mainly *malicious*, *questionable*, *unknown*, and *websites with adult content*. This ratios indicate that, despite, that web tracking by itself cannot be used to discriminate the maliciousness or greyness, surprisingly, *malicious* or *grey* websites tend to only include web tracking scripts and not other functionality.

The relation between websites with some or only tracking scripts with *Webutation* (see Table 4) hinted that the presence of only tracking scripts is correlated to the reputation of a website. 15.41% of the websites in the *Red* category and 15.31% in the *Yellow* categories only used tracking scripts, while only the 6.45% and 5.95% of the *Grey* and *Green* categories did, respectively. Since users do not tend to think about web tracking and the high presence of web tracking in every sort of website, we believe that this results indicate that websites perceived as *bad* by users' have a higher ratio of web tracking than non-tracking, similarly as happened with the websites category.

Table 5: 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*

Domain Demographics

We also measured domains used to host tracking scripts, non-tracking scripts, and both tracking and non-tracking scripts (see Table 5). The reason to not only analyze websites but also domains, is that we seek to analyze both the usage and also the provision of web tracking in the web. In fact, previous work in web vulnerabilities found the relevance of the domains hosting scripts with the ultimate nature of them [19].

We discovered that, similarly as happened to websites using web tracking scripts, domains usually host web tracking scripts alongside non-tracking scripts. Indeed, the percentage of them hosting solely tracking scripts is not negligible (10.54%).

With regards to the relation of countries with their domains, despite the fact that small countries and tax havens surprisingly appeared in the list with the highest number of domains hosting only tracking scripts, the small number of domains in those cases render it as a not conclusive finding.

Likewise, we found several cases of either AS owners, ASNs, or ISPs whose domains were only used to host tracking scripts but given their small number of domains in each of the cases, we consider these results irrelevant.

As we did when studying websites, we analyzed the correlation between the *webutation* of the domain and its hosting of tracking scripts (see Table 6). We corroborated that, as happened with websites, the presence of only tracking scripts in domains had an effect in the reputation: *Yellow* and *Red* were composed of a 22.87% and 23.71% of the domains hosting only tracking scripts while *Grey* and *Green* categories only

Table 6: Relation between *webutation* (by the reputation online service) and domains hosting some or only tracking scripts. *% Some T.* means the percentage of domains hosting some tracking scripts and *% Only T.* means the percentage of domains that only host tracking scripts.

Category	**# Domains**	**% Some T.**	**% Only T.**
Red	5,898	95.07%	22.87%
Yellow	10,447	95.83%	23.71%
Grey	447,687	94.05%	11.23%
Green	465,007	96.99%	9.44%

Table 7: Known tracking detection comparison between blacklisting, code hashing, and TRACKINGINSPECTOR that detected the 100% of the known tracking scripts.

Solution	**# Scripts**	**% Known Sc.**
Blacklisting (scripts)	1,068,652	43.80%
Blacklisting (domains)	825,617	33.84%
Blacklisting (all)	1,577,251	64.65%
Code hashing	49,807	2.04%

contained 11.23% and 9.44%. Taking into account how *webutation* works, the results may be due to negative user ratings aware of tracking techniques or also, because the external tools determine it as malicious. In both cases, it seems likely that malicious or suspicious websites (determined either by users or services) tend to host only tracking scripts and not a combination of web tracking and non-tracking scripts.

3.3 Analysis of Known Tracking

To compare the detection capabilities of TRACKINGINSPECTOR with current web tracking blocking solutions, we measured the number of known script samples that blacklisting solutions would have blocked. From all the script blacklisting mechanisms (e.g., script name or script complete URL), we chose script name blacklisting as the baseline because it provided a broader detection than the other alternatives. In this way, we computed the number of known tracking script samples whose name was the same as the one stored in the *Script Database*. The blacklisted domains were also used in order to measure their blocking capability. Another possible solution not used in web tracking detection but widely used in other domains has also been explored: code hashing: we measured the number of scripts whose code exactly matches scripts in the *Script Database*. Results (shown in Table 7) show that script and domain blacklisting captured 43.80% and 33.84% of the known tracking script versions, respectively. Combined blacklisting solutions would had blocked the 64.65% of the known tracking script samples while code hashing only would had captured 2.04% of the samples. These results indicate that current anti-tracking solutions are clearly not enough, not only to fight against completely unknown tracking scripts, but they cannot deal with known tracking scripts versions whose code has been modified and nearly the half of all the script samples in the wild would not have been blocked by them.

Moreover, we measured the prevalence of samples and versions of the tracking scripts stored in the *Script Database*. In particular, *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 popular scripts we can find: 20.92% regarding advertisement companies (*33Across*, *Pzyche*, and *QuantCast*), 3.49% from analytics companies (*Yandex Metrica* and *comScore*), and 2.00% social analytics samples (*FlCounter* and *Pinterest*). These known tracking script samples were used by the 84.02% of the websites with scripts (see Table 8).

Table 8: Known tracking prevalence in websites.

# websites	770,440
– % in websites with tracking	86.10%
– % in websites (considering sites with scripts)	84.02%

Table 9: 10 most popular top-level domains hosting known tracking script samples.

Domain	**# Websites**
`google-analytics.com`	608,223
`google.com`	144,853
`googlesyndication.com`	140,260
`ytimg.com`	60,373
`yandex.ru`	47,805
`doubleclick.net`	44,509
`ajax.googleapis.com`	40,610
`scorecardresearch.com`	36,297
`googleadservices.com`	34,770
`googletagservices.com`	33,303

The known scripts samples were hosted in 540,369 top-level domains. The 10 most popular domains are shown in Table 9, beings `google-analytics.com` was the most popular domain. Their scripts were present in 63.14% of the websites with scripts. The rest of the domains belonged to Google or to well-known advertisement services. As it can be noticed, the hosting of known tracking scripts follows a long-tail distribution with a small number of domains (or companies) hosting the most number of script downloads , confirming the long-tail nature of web tracking provision regarding known tracking script samples.

3.4 Analysis of Unknown Tracking

Regarding the *Unknown Tracking Analysis*, TRACKINGINSPECTOR flagged 79.64% of all the tracking script samples as likely unknown tracking candidates. 41.11% of them corresponded to samples downloaded from blacklisted domains. The remaining 5,621,019 likely tracking script candidates were not previously known by any solution. Due to the high prevalence of unknown tracking scripts, we analyzed them separately to understand their nature.

Unknown Tracking Analysis In-the-wild Evaluation

TRACKINGINSPECTOR's *Unknown Tracking Analysis* component flagged more than 8,000,000 of scripts as unknown tracking candidates and more than 5,000,000 were not previously known by any blacklist or solutions present. Even thought we already performed a 10-fold cross validation of the *Script*

Table 10: 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 11: 10 most popular blacklisted top-level domains hosting unknown tracking script samples.

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

Database, due to the large number of scripts discovered, we decided to perform an additional in-the-wild manual validation to measure the real capacities of our technique and to obtain a more real-wold measurement about our tool's capabilities.

To this end, we extracted a random sample from the ones marked as potentially unknown tracking scripts in the large-scale analysis. The sample is composed of 273 scripts, representing a 90% confidence level and a ±5% confidence interval. By manually analyzing these samples, we discovered that 85.35% are directly involved in some type of web tracking, although nearly all of them showed suspicious behaviours (e.g., checking different data from the browser). These results are in the same line as the ones that our tool obtained in the 10-fold cross validation described in Section 2. Since the goal of this component is to discover new tracking scripts for the database and it is not intended to be used as a client-side detector, we believe that the percentage of FP is more than acceptable for this task.

Unknown Tracking in Blacklisted Domains

Unknown tracking scripts downloaded from blacklisted domains appeared in 70.49% of the websites with scripts (see Table 10). Although we cannot consider that all the scripts hosted in blacklisted domains as web tracking, the fact that *Unknown Tracking Analysis* detected them, suggested its discovery capabilities.

Surprisingly, only 7,455 blacklisted domains (see Table 11 for details about the 10 most prevalent ones) hosted 3,923,615 of this type of unknown tracking script samples. This number of domains is much smaller than in the case of known or completely unknown tracking scripts. This fact demonstrates the relevance of these blacklisted domains in the web tracking ecosystem. In particular, script samples from `facebook.com` was present in 18.42% of the websites with scripts.

Table 12: Completely unknown tracking prevalence in websites.

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

Table 13: 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 14: 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

New Unknown Potential Tracking

Scripts & Domains

In relation to completely unknown tracking candidates, they represented 58.89% of the likely completely unknown tracking samples flagged by the *Unknown Tracking Analysis* component. The script presence in websites varied with regards to whether the domain was blacklisted or not. In the case of completely unknown tracking script candidates, their prevalence was higher than in the case of blacklisted domains: 90.69% of the websites with scripts used likely unknown tracking (see Table 12), corroborating again the low detection coverage of current anti-blocking solutions.

Due to the high number of discovered scripts, we performed an additional analysis to comprehensively understand the nature of these scripts. 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 of the same script through code hashing and measured the number of unique scripts in each domain type (see Table 13). The typical way of using web tracking is with third-party domains, but we found that the number of websites that use their their own domain is significant in the case of unknown potential tracking scripts. In particular, most of them (73.75%) were embedded in the HTML, that can be used to bypass current blacklisting solutions.

We also found that 57.73% of the completely unknown

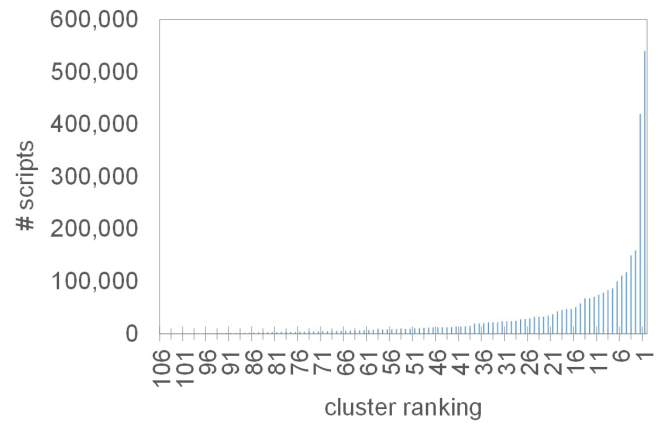

Figure 2: Distribution of the unknown tracking candidates in the computed known tracking clusters.

Table 15: Popular clusters per domain type.

Domain	Cluster	% Scripts
HTML	Downloading	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%

scripts were unique by hash. TrackingInspector also discovered a relative small number of unique scripts whose samples were repeated across different domain types, specially among first and third party domains. This indicates that using scripts in different domain types is not a common practice.

We computed the presence of the top-level domains hosting unknown tracking candidates in websites. Due to their distribution in the different domain types, we focused only on the scripts downloaded from third-party domains. The 10 most popular domains (shown in Table 14) that hosted previously unknown tracking candidates were different types of domains such as CDNs, search engines, social networks, gambling companies, or advertisement companies.

Clustering

In order 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 unknown tracking candidates, we conducted a clustering analysis in two different steps. The first step consisted in clustering the 957 known tracking scripts stored in our *Script Database* to find the different known script categories/types. We chose the *Affinity Propagation* clustering algorithm [21] due to its capability of automatically computing the cluster membership of an unknown sample and because it does not need to specify the number of clusters. As result, the known scripts were distributed in 106 different clusters.

In the second step, we computed the closest cluster for each of the different unknown tracking script candidates through the previously calculated clusters (see Figure 2 for the distribution of unknown tracking candidates in the clusters). The most popular category contained *downloading* scripts and comprised 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 analyzed the clusters with regards to the type of domain that hosted the scripts (see Table 15). We discovered that scripts hosted in first-party and third-party domains behaved similar, whereas HTML-embedded domains did not.

4 Case Studies

4.1 Script Renaming Techniques

Script blacklisting techniques only blocked 43.80% of the known scripts or versions TrackingInspector detected. Since *script renaming* is an effective technique to circumvent current blacklisting solutions, we performed a study of this phenomenon to deepen in its understanding.

More than 35% of the samples of each original scripts changed their name. Each original script behaved differently: while samples of 200 of the original scripts did not present any modification, versions from other 95 original scripts always changed their name. Therefore, we analyzed in more detail three particular scripts to discover different *script renaming* techniques: `evercookie.js` [31], `piwik.js` [32], and `dota.js` [33]. In particular, 58.33% of the samples of `evercookie.js`, 91.13% of the versions of `piwik.js`, and 99.66% of `dota.js` modified their original script name.

Among the script renaming techniques, the following categories can be established: (i) *related script renaming*, (ii) *random/neutral script renaming*, (iii) *functionality script renaming*, and (iv) *misleading script renaming*. *Related script renaming* changes the script name to one that is 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 of the original to an apparently random name, such as `penguin2.js`, `scripts3.js`, and `welcome.js`. *Functionality script renaming* modifies the name describing the internal goal or functionality of the script. For instance, several scripts changed the original script name to `fingerprint.js`, and `tracking.js`. Finally, *misleading script renaming* possibly bypasses blacklisting by modifying the original script name to a well-known one such as `jquery.alt.min.js` and `j.min.js`.

4.2 Canvas and Font Fingerprinting

Canvas fingerprinting [3] and font probing [2] are two types of stateless device fingerprinting. Due to their relevance and the concern they raise, we decided to study them separately and determine how many unknown scripts of this type of fingerprinting TrackingInspector discovered.

[31] `piwik.js` is a well-known open-source analytics service.
[32] `evercookie.js` bypasses the cookie storage policy of the browser.
[33] `dota.js` performs canvas fingerprinting (as reported in [13]).

Table 16: 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*

Table 17: Website prevalence of top 3 potential unknown device fingerprinting scripts.

Script	Domain	# Websites
	Font probing	
`buttons.js`	`sharethis.com`	18,625
`ice.js`	`infolinks.com`	5,333
`loaded.js`	`acexedge.com`	110
	Canvas fingerprinting	
`Admeta.js`	`atemda.com`	981
`sat.js`	`myswitchads.com`	514
`bs-engine.js`	`bshare.cn`	163
	Font & Canvas fingerprinting	
`image.js`	`magnuum.com`	131
`fp.min.js`	`adtarget.me`	31
`ax.js`	`gmyze.com`	31

We implemented two small programs: one for the detection of font probing and another one for detecting canvas fingerprinting scripts. These programs were designed to filter the unknown potential tracking script samples that match rules for font probing and for canvas fingerprinting. In this way, we built two basic set of static rules based on the scripts discovered by [12] and [13]. It is important to note, that these rules are not detection methods by themselves but a filtering technique for scripts that have already exhibited web tracking or fingerprinting behavior. 710 unknown unique potential device fingerprinting scripts were found through these methods: 408 exhibited canvas fingerprinting behavior, 247 font probing behavior, and 55 showed both behaviors.

Since blacklisted domains may have blocked some of versions of these unknown device fingerprinting scripts, we used these new potential device fingerprinting scripts as the *Script Database* and performed a *Known Tracking Analysis* to measure the prevalence of their samples and versions in the Alexa top 1M as well as the domains used for hosting them (see Table 16). 28,632 samples were detected and 27,818 websites used these unknown scripts.

We also analyzed the top potential device fingerprinting unknown scripts for each type (see Table 17 for the top 3 scripts per fingerprinting type). The most prevalent unknown font probing script was `buttons.js`. This script was hosted by *sharethis.com*, a well-known social widget for sharing content in different social networks. In the case of canvas fingerprinting, the most used unknown script was `Admeta.js` that was downloaded by 981 websites from *atemda.com*, an ad exchange provider. In the case of scripts containing both canvas and font probing techniques, `image.js` was the most prevalent unknown script, being downloaded by 131 websites from the domain `magnuum.com`, a content delivery network.

```
<iframe
    name="z4Pdb4sl"
    src="http://202.172.54.119/
       jquery.min.js"
    width="400" height="400"
    style="position: absolute;
    left: -9999em;">
</iframe>
```

Figure 3: The `iframe` used to download the malicious script.

4.3 Fingerprinting-driven Malware

The goal of this paper was to understand the current landscape of web tracking and not to study its potential relation with targeted malware. However, recent reports [7,8] linked targeted malware campaigns with a previous fingerprinting step and also according to our own results, websites using only tracking tend to be more prevalent in questionable categories.

Therefore, we inspected domains hosting only tracking scripts and discovered very suspicious scripts. For example, a script started by performing a fingerprinting step that included identification of the browser, *Java*, *Flash Player*, *SilverLight*, and checked the presence of a Chinese antivirus. Then, it performed very suspicious calls to conduct a likely malicious behavior. 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 fingerprinting-driven malware detection technique.

This script was hosted in two IPs: `101.99.68.18` and `202.172.54`. The server `101.99.68.18` was allocated in the ISP *Piradious-NET* and the server `202.172.54` was hosted in *M1 Connect Pte. Ltd.*, both of them known for some cases of malware hosting. The script was named `jquery.min.js`, posing as a well-known script of the benign library *jQuery*. Besides this obvious hiding effort, the script, surprisingly, was not obfuscated and its code was very clear. Two Chinese websites in the Alexa top 1M (`52life.cc` and `examres.com`) used this script. Each website contained a different iframe sharing the same name (one of them is shown in Figure 3) that was used to download the malicious script.

By searching the name of the iframes, we discovered that this script was part of the *Chinad* botnet, as reported by *MalwareBytes* [22, 23]. We did not perform a manual malware analysis, since it has already performed by *MalwareBytes*. As explained in their report, this script was a exploit kit that compromised Chinese websites to fingerprint users looking for vulnerable installed components to later exploit vulnerabilities in *Java* (CVE-2011-3544 and CVE-2012-4681), *Internet Explorer* (CVE-2014-6332), and *Flash* (CVE-2015-0311) and download versions of the *Chinad* botnet. This malware was apparently oriented to perform DDoS attacks.

To the best of our knowledge, the websites where the exploit kit was discovered had not been previously documented as infected. In fact, it is important to note that the goal of this paper is not study malicious scripts and, therefore, we did not explore this phenomena beyond this interesting case study.

5 Discussion

After presenting the results of our large-scale study on web tracking on the Internet, we discuss here our findings regarding the raised research questions, including the implications and limitations of our results.

How widespread is web tracking on the Internet?

Our results indicate that web tracking is very frequent in popular websites. In fact, we believe that web tracking is such a big part of the web, that it is going to be hard to limit or change.

We discovered that more than 40% of the analyzed samples correspond to previously unseen potentially tracking scripts. After analyzing the differences between the known tracking script samples and the unknown ones, we also found that the variability among known and unknown tracking scripts varied: while known tracking samples are versions of an original script, unknown samples tend to be more unique. Since the most popular scripts can also be considered the most invasive ones, currently known tracking samples pose a more invasive threat. However, the discovered large number of unknown potential tracking samples is concerning and, therefore, we think that further efforts should be done in version and unknown behaviors detection.

Advertising companies have already stated that web tracking is required for the web economy [6]. However, other techniques have been proposed for analytics and targeting preserving users' privacy (e.g., [24–29]). We consider that this line of work may overcome privacy-invasive techniques for web tracking and reduce their usage, while preserving the core functionality of web advertisement.

What is the current ecosystem in web tracking provision and deployment on the Internet?

In our study, questionable categories (e.g., *phishing*, *spam*, *malware*) tend to present a higher number of websites using only tracking scripts. By further exploring these results, we discovered several IPs that only hosted tracking scripts within the domains. In two of them, TRACKINGINSPECTOR discovered the fingerprinting behavior of a script that was part of a targeted *fingerprinting-driven malware* campaign directed to Chinese websites previously reported by *MalwareBytes* [22, 23]. This script was found in two additional websites that were not reported as infected. We believe that this type of fingerprinting-driven and targeted malware campaigns may become an important concern.

These results also show that the tracking provision distribution varies depending on the script being previously known, hosted in a blacklisted domain or completely unknown. While known tracking scripts are mainly hosted in *Google* or analytics/advertisement companies, unknown tracking scripts are hosted in different domains types. One relevant finding is that the well-known *ShareThis* service resulted to host a font probing script. We also found that network entities such as the AS owner, ASN and ISP or the country where they are hosted do not seem to have any influence on tracking, but reputation does.

The correlation between domains or websites using only tracking scripts and questionable categories or low reputation, can be explained because malicious activities do not usually exhibit additional functionalities besides the ones oriented towards their endeavors.

Can current blacklisting solutions limit or block most web tracking on the Internet?

After performing a comparison between our solution with blacklisting solutions (both at domain and script level), only 43.8% of the known scripts detected by TRACKINGINSPECTOR, would have been blocked by script blacklisting and 33.84% by domain blacklisting (64.65%, combining both solutions). Regarding to unknown potential tracking scripts, just 41.11% of the scripts were hosted in blacklisted domains.

These blacklisting solutions were not able to capture the majority of the web tracking scripts. Indeed, in one of our case studies, we analyzed the name variations that circumvent script blacklisting, discovering that it is an extended practice among web tracking scripts. Therefore, we believe that current blacklisting solutions should be extended or substituted by other approaches such as the one presented here for *Known Tracking Analysis*.

Can TRACKINGINSPECTOR *discover new web tracking scripts on the Internet? To what extent?*

TRACKINGINSPECTOR discovered more than 5.5 million completely unknown candidate tracking scripts. By analyzing a random sample of these scripts we further evaluate the discovery capability, with a result of more than 85% true positive ratio.

More than 700 new potential device fingerprinting unique scripts were discovered, a number higher than any reported result. These device fingerprinting scripts were present in many websites and domains, including well-known services.

We believe that the findings of TRACKINGINSPECTOR are enough to demonstrate its capability of discovering new web tracking scripts. However, we also think that there are two major limitations in the presented approach. Firstly, our tool is highly reliant on previously known web tracking scripts for both known and unknown tracking analysis. Despite the fact that this approach results in a framework easier to generalize and to upgrade than rule-based ones, it also represents a problem when updating the *Script Database*. To overcome this limitation, both components of TRACKINGINSPECTOR should be used for different but complementary tasks. *Known Tracking Analysis* would detect variations of known tracking scripts and *Unknown Tracking Analysis* would gather websites flagging any possible tracking script candidate. These samples would be further analyzed to confirm their nature and then added to the *Script Database*. Secondly, *Unknown Tracking Analysis* may flag erroneously scripts that, even they exhibit fingerprinting behavior, they are not intended for web tracking but for personalization or similar tasks. To reduce these errors, our tool may be combined with a whitelisting approach.

6 Related Work

Due to the concern that web tracking raised to users' privacy, a hectic research line emerged in the past years dedicated to analyze and block these techniques. Researchers themselves created new web tracking methods using several techniques such as fonts and browser metrics [2], measurement of timing metrics [30,31], the JavaScript engine [32], the use of the rendering engine [33], the clock skew [34], or the HTML5 Canvas API [3].

One of the first web tracking analysis that included HTML cookies was the influential work performed in [35]. Following this work, Mayer & Mitchell [36] studied the different techniques for web tracking including their policies and developed a tool to measure web privacy. Several more recent works have studied the presence of different forms of web tracking on the web. Roesner et al. [9] presented a taxonomy for third-party tracking using cookies, measuring their presence in the web. Nikiforakis et al. [10] studied three known fingerprinting companies and discovered 40 websites within the Alexa top 10,000 sites using techniques such font probing. Acar et al. [12] discovered 404 sites in the top million using JavaScript-based fingerprinting and 145 sites within the Alexa top 10,000 sites using Flash-based fingerprinting. In another work, Acar et al. [13] found a 5% prevalence of canvas fingerprint in the Alexa top 1M web 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 [19] determined the important 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 Very recently, an unpublished parallel work [14] 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 sophisticated device fingerprinting techniques. A recent work [4] has shown the capability of 17 attributes to build technique of fingerprinting and have shown that it can used with all modern browsers, including mobile ones. In also recent work, Lerner et al. [11] presented *TrackingExcavator* and performed a retrospective analysis of how tracking has evolved since 1996. Their findings include that third-party has increased over time and how the most important trackers have become popular over time.

Compared with these previous contributions, our work differs in many ways. First, it is committed to a truly generic web tracking detection rather than to specific techniques. TrackingInspector does not depend on blacklists or specific rules to detect tracking scripts, but on the previous known tracking scripts. Second, through our method we have been capable of detecting any type of tracking in our large-scale study, discovering more than 3 million new unique tracking script candidates, a number higher than any reported previous work. We also discovered 710 new potential device fingerprinting scripts, including 408 canvas fingerprinting scripts, 247 font probing scripts, and also 55 completely new scripts that exhibited both fingerprinting behaviors, a number also higher than previous work. Our tool also automatically detected the fingerprinting behavior of an already known targeted malware campaign for the first time in the literature.

7 Conclusions

In this paper we presented the first large-scale study of generic web tracking on the web. Existing tools are limited in the detection of web tracking. Therefore, we developed TrackingInspector, a generic web tracking script detector to identify known tracking script samples and flag unknown tracking script candidates.

Using our method, we measured the web tracking prevalence in websites and their providing domains as well as web tracking ecosystem. The results show that web tracking is very extended and that current solutions cannot detect every known or unknown tracking scripts. In addition, we examined the hiding techniques used to avoid blacklists, determining different script renaming techniques. TrackingInspector detected both known or variations of tracking scripts and discovered likely unknown web tracking candidates. We also found new potential stateless device fingerprinting scripts and measured their prevalence, showing that even well-known companies provide these type of scripts. Among the discovered unknown web tracking scripts, we found a previously reported malware campaign that targeted Chinese websites, showing that malicious activities sometimes exhibit fingerprinting behavior that TrackingInspector can automatically detect. We also believe that *fingerprinting-driven malware* may become an relevant problem in the future.

References

[1] Ashkan Soltani, Shannon Canty, Quentin Mayo, Lauren Thomas, and Chris Jay Hoofnagle. Flash cookies and privacy. In *Proceedings of the AAAI Spring Symposium: Intelligent Information Privacy Management*, volume 2010, 2010.

[2] Peter Eckersley. How unique is your web browser? In *Proceedings of the Privacy Enhancing Technologies (PETS)*. Springer, 2010.

[3] Keaton Mowery and Hovav Shacham. Pixel perfect: Fingerprinting canvas in HTML5. *Proceedings of the Web 2.0 Workshop on Security and Privacy (W2SP)*, 2012.

[4] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. In *Proceedings of the IEEE Symposium on Security and Privacy (Oakland)*, 2016.

[5] M Ayenson, DJ Wambach, A Soltani, N Good, and CJ Hoofnagle. Flash cookies and privacy II: Now with HTML5 and Etags respawning (2011). *Social Science Research Network Working Paper Series*, 2011.

[6] Natasha Singer. 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.

[7] Security Response, Symantec. The Waterbug attack group. `http://www.symantec.com/content/en/us/enterprise/media/security_response/`

whitepapers/waterbug-attack-group.pdf, 2015.

[8] Threat Intelligence, FireEye. Pinpointing Targets: Exploiting Web Analytics to Ensnare Victims. https://www2.fireeye.com/rs/848-DID-242/images/rpt-witchcoven.pdf, 2015.

[9] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. Detecting and defending against third-party tracking on the web. In *Proceedings of the USENIX conference on Networked Systems Design and Implementation (NDSI)*, 2012.

[10] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Proceedings of IEEE Symposium on Security and Privacy (Oakland)*, 2013.

[11] Adam Lerner, Anna Kornfeld Simpson, Tadayoshi Kohno, and Franziska Roesner. 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.

[12] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. FPDetective: dusting the web for fingerprinters. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2013.

[13] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2014.

[14] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. http://randomwalker.info/publications/OpenWPM_1_million_site_tracking_measurement.pdf, 2016. [Draft].

[15] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[16] Hans Peter Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):309–317, 1957.

[17] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[18] Tin Kam Ho. Random decision forests. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 1995.

[19] Nick Nikiforakis, Luca Invernizzi, Alexandros Kapravelos, Steven Van Acker, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. 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)*.

[20] Charlie Curtsinger, Benjamin Livshits, Benjamin G Zorn, and Christian Seifert. ZOZZLE: Fast and Precise In-Browser JavaScript Malware Detection. In *Proceedings of the USENIX Security Symposium (Sec)*, 2011.

[21] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

[22] 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/.

[23] MalwareBytes. Unusual exploit kit targets chinese users (part 2). https://blog.malwarebytes.org/intelligence/2015/06/unusual-exploit-kit-targets-chinese-users-part-2/.

[24] Vincent Toubiana, Arvind Narayanan, Dan Boneh, Helen Nissenbaum, and Solon Barocas. Adnostic: Privacy preserving targeted advertising. In *Proceedings of the Network and Distributed System Symposium (NDSS)*, 2010.

[25] Saikat Guha, Bin Cheng, and Paul Francis. Privad: practical privacy in online advertising. In *Proceedings of the USENIX conference on Networked Systems Design and Implementation (NDSI)*, 2011.

[26] Matthew Fredrikson and Benjamin Livshits. Repriv: Reimagining content personalization and in-browser privacy. In *Proceedings of the IEEE Symposium on Security and Privacy (Oakland)*, 2011.

[27] Mikhail Bilenko, Matthew Richardson, and Janice Tsai. Targeted, not tracked: Client-side solutions for privacy-friendly behavioral advertising. In *Proceedings of the Privacy Enhancing Technologies (PETS)*, 2011.

[28] Michael Backes, Aniket Kate, Matteo Maffei, and Kim Pecina. Obliviad: Provably secure and practical online behavioral advertising. In *Proceedings of the IEEE Symposium on Security and Privacy (Oakland)*, 2012.

[29] Istemi Ekin Akkus, Ruichuan Chen, Michaela Hardt, Paul Francis, and Johannes Gehrke. Non-tracking web analytics. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CSS)*, pages 687–698. ACM, 2012.

[30] Keaton Mowery, Dillon Bogenreif, Scott Yilek, and Hovav Shacham. Fingerprinting information in javascript implementations. *Proceedings of the Web 2.0 Workshop on Security and Privacy (W2SP)*, 2011.

[31] Tom Van Goethem, Wouter Joosen, and Nick Nikiforakis. The clock is still ticking: Timing attacks in the modern web. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CSS)*, 2015.

[32] Martin Mulazzani, Philipp Reschl, Markus Huber, Manuel Leithner, Sebastian Schrittwieser, Edgar Weippl, and FC Wien. Fast and reliable browser identification with javascript engine fingerprinting. In *Proceedings of the Web 2.0 Workshop on Security and Privacy (W2SP)*, 2013.

[33] Thomas Unger, Martin Mulazzani, Dominik Fruhwirt, Marco Huber, Sebastian Schrittwieser, and Edgar Weippl. Shpf: Enhancing http (s) session security with browser fingerprinting. In *Proceedings of the 8th International Conference on Availability, Reliability and Security (ARES)*, 2013.

[34] Tadayoshi Kohno, Andre Broido, and Kimberly C Claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2):93–108, 2005.

[35] Balachander Krishnamurthy and Craig Wills. Privacy diffusion on the web: a longitudinal perspective. In *Proceedings of the International Conference on World Wide Web (WWW)*, 2009.

[36] Jonathan R Mayer and John C Mitchell. Third-party web tracking: Policy and technology. In *Proceedings of the International Symposium on Security and Privacy (Oakland)*, 2012.