Detection, characterization, and countermeasure of first-party cooperation-based third-party web tracking

by

DAO THI THU HA

Dissertation

submitted to the Department of Informatics

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

The Graduate University for Advanced Studies, SOKENDAI
September 2022

Committee

Advisor	Dr. Kensuke Fukuda Associate Professor of National Institute of Informatics/SOKENDAI, Japan
Subadvisor	Dr. Yusheng Ji Professor of National Institute of Informatics/SOKENDAI, Japan
Examiner	Dr. Megumi Kaneko Associate Professor of National Institute of Informatics/SOKENDAI, Japan
Examiner	Dr. Shunji Abe Associate Professor of National Institute of Informatics/SOKENDAI, Japan
Examiner	Dr. Michihiro Koibuchi Professor of National Institute of Informatics/SOKENDAI, Japan
Examiner	Dr. Daisuke Miyamoto Associate Professor of the University of Tokyo, Japan

Acknowledgements

First and foremost, I would like to thank my supervisor, Professor Kensuke Fukuda, for his invaluable mentorship. He welcomes me to his laboratory and provides me the opportunity to work in a unique research environment during a five-month internship and over the past three years of study. I owe a lot of my professional growth to his guidance. He taught me numerous and constructive advice to succeed in my research. He also spent precious time with me for discussions and corrected my articles to make them acceptable for submission. Thank you for always believing in me, even during the moments when I doubted myself!

I would like to express my special thanks to my mentor and also my collaborator, Johan Mazel, who has great support and always give me helpful comments throughout my research progress. I am incredibly lucky to have worked with him. In addition, he spent many hours proofreading each manuscript on time, ensuring its content meets the requirements for submission. The research in this dissertation would not have been possible without his contribution.

I am also grateful to my dissertation committee, Professor Yusheng Ji, Professor Shunji ABE, Professor Megumi Kaneko, Professor Michihiro Koibuchi, and Professor Daisuke Miyamoto, for their valuable advice and insightful comments on my research progress.

I thank the National Institute of Informatics (NII) and The Graduate University for Advanced Studies (SOKENDAI) for funding my PhD research. I would also like to thank our Lab secretaries, Mr. Kunikazu Yoda and Ms. Yuumi Asanuma; and the NII/SOKENDAI secretaries, for all the help and support they have given me over the years. Also, I want to thank my lab members, Guannan Hu, Satoru Kobayashi, and Ryusei Shiiba. They provided invaluable critical feedback and a friendly environment

for me to learn and excel. Thank you to my best friends in Tokyo, Ly Dinh, An Le, An Dao, and Hong Vuong, for their unwavering support and patience through the ups and downs of grad school. My stay in Japan was really enjoyable thanks to the nice people I met in NII and they became invaluable friends in my whole life. Thank you to my wise friend, Dang Huynh. You have inspired me to be better person without making me feel inadequate. It would be impossible to list all you have done to encourage me through this challenging time.

I express my deepest and most sincere gratitude to my best friends and my special friend for their unconditional love and encouragement. Thank you for always being by my side and taking care of me. Finally, I am immensely grateful to my parents and all my family members in Vietnam. Thank you for giving me the freedom to do whatever I want to do. Your love and support throughout my life are the keys of all my achievements.

Abstract

Third-party web tracking has been used for collecting and correlating user browsing behavior. It is becoming more and more ubiquitous, thus this brings an increase in privacy concerns from Internet users. Due to the increasing use of ad-blocking and third-party web tracking protections, tracking providers have introduced new techniques to continue maximizing their profit based on user data. As the recent sophisticated techniques, the third-party tracking providers have leveraged cooperation from the first-party for tracking user activities. Thus, in this dissertation, we focus on the first-party cooperation-based third-party web tracking, including CNAME cloaking-based tracking and PII leakage-based tracking. In particular, third parties have leveraged cooperation from the first-party by using first-party subdomain Canonical Name Record or Alias (CNAME) record in the Domain Name System (DNS), to bypass the filter lists in browsers and extensions that disguise requests to a third-party tracker as first-party ones; they also have leveraged cooperation from the first-party by using user's personally identifiable information (PII) of first-party authentication flows, to create an identification that is a persistent identity. The goals of this dissertation are to perform a first in-depth analysis of the first-party cooperation-based third-party web tracking and develop the countermeasures to protect user privacy against these tracking techniques on the Internet.

In the first half of this dissertation, we detect, characterize, then develop a countermeasure to protect the end-user against the first-party cooperation-based third-party web tracking technique, namely CNAME cloaking-based tracking. This technique misleads web browsers into believing that a request for a subdomain of the visited website originates from this particular website, while this subdomain uses a CNAME to resolve to a tracking-related third-party domain. It thus circumvents the

third-party targeting privacy protections. Specifically, we first characterize CNAME cloaking-based tracking by crawling the top pages of the Alexa Top 300,000 sites and analyzing the usage of CNAME cloaking with CNAME blocklist. We also point out that browsers and privacy protection extensions are largely ineffective to deal with CNAME cloaking-based tracking except for Firefox with a developer's version of the uBlock Origin extension. Secondly, we propose a supervised machine learning-based approach to detect CNAME cloaking-based tracking without the on-demand DNS lookup. We show that the proposed approach outperforms well-known tracking blocklists. Finally, to circumvent the lack of DNS API in Chrome-based browsers, we design and implement a prototype of the supervised machine learning-based browser extension to detect and filter out CNAME cloaking tracking, called *CNAMETracking Uncloaker*. Our evaluation shows that *CNAMETracking Uncloaker* is able to filter out CNAME cloaking-based tracking requests without performance degradation when compared with the vanilla setting on the Chrome browser.

In the second half of this dissertation, we detect, characterize, then develop a countermeasure to protect the end-user against the first-party cooperation-based third-party web tracking technique, namely PII leakage-based tracking. This technique uses personally identifiable information (PII) to perform cross-site, cross-browser, and cross-device tracking. We document a PII-based tracking ecosystem that leverages user sign-up and sign-in flows on the popular shopping sites from the Tranco Top 10,000 sites. We perform a first in-depth analysis of PII leakage and present a previously unknown persistent web tracking technique based on this data transfer, which enables tracking providers to generate and store a unique persistent identifier for a user on their servers. By measuring the presence of Online Behavioral Advertising (OBA), we confirm that the tracking providers use leaked PII in their advertising strategies for cross-site, cross-browser, and cross-device targeting and personalization. Also, to provide a wider picture of current in-browser privacy protection techniques, we evaluate the effect of browsers and well-known blocklists against PII leakage. Finally, we propose a hybrid approach to detect PII leakage by combining heuristic and supervised machine learning approaches. We show that the proposed approach outperforms well-known tracking blocklists.

We conclude by emphasizing the research contributions made by this thesis and present some open research problems. We first highlight the practical implication of

our work to researchers, browser vendors, and Internet users. We think that this work will stimulate follow-up works in the research community and lead to web browser improvements. We also think that this work increases Internet user awareness regarding privacy. In addition, we identify a number of possible research directions, including measurements, perspectives, and recommendations to improve transparency on the World Wide Web.

Keywords: Privacy,CNAME cloaking-based tracking, third-party web tracking, machine learning techniques, countermeasure, browser extension, PII leakage, PII leakage-based tracking.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivating problems

World Wide Web (more commonly known as WWW, W3, or the Web) was invented by Tim Berners-Lee, a British scientist in 1989 for automated information-sharing between scientists in universities and institutes around the world [1]. It has opened up the internet to everyone, not just scientists. It connects people in the world and allows people to share their work and thoughts through personal blogs, community websites, social networking websites, and more. To provide a good user experience, website owners have used many tracing techniques by the first-party domain (a domain the user is visiting directly), such as first-party cookies, to collect analytics data, remember shopping cart information, retain language settings, and perform other useful functions.

But user tracking did not stop at the first-party level. It provided the means for a revolution in the history of marketing. Marketing products and services to Internet users has become a key economic driver in the Internet economy and generates

revenue for a wide variety of websites and services [2]. To get a fuller understanding of user interests, tracking users across websites, browsers, and devices would be very profitable by providing advertising that focuses on the specific traits, interests, and preferences of a user. Third-party web tracking techniques, which refer to the practice of an entity other than the domain directly visited by the user, have been developed for this purpose. It allows them to personalize user content accordingly across sites, browsers, and devices. The large-scale collection and analysis of personal information even constitute the core business of many online companies. Since the world's first online advertising technology company, namely DoubleClick, started to exploit third-party cookies to track users across the web in 1995, third-party web tracking has been becoming sophisticated over time. It relies on a wide variety of web tracking technologies, ranging from stateful tracking mechanisms that recognize users by retrieving information stored on user's machines, and stateless tracking mechanisms that recognize users without storing any information, such as HTTP cookies, evercookies, and HTML5 fingerprinting [3].

Despite the much-attended successes in utilizing the Internet marketplace using third-party web tracking, one of the major impediments against full-scale integration of the Internet marketplace with modern business is the privacy issue. Internet users even perceive fear and distrust regarding the loss of personal privacy in the online environment [4]. In a TRUSTe study, 92% of British Internet users worry about their online privacy [5]. This concern has resonated with regulators and web browser vendors. In 2018, the General Data Protection Regulation (GDPR) has set out rules for explicitly gathering user consent regarding targeted advertising within the European Union [6]. Similarly, the California Consumer Privacy Act (CCPA) regulations in California aim to protect the data of users within the state [7]. Browsers and extensions already block known third-party tracking. For instance, Safari and Firefox browsers already block the most common tracking technique, third-party cookies [8, 9], and Chrome also announced that they will block third-party cookies in 2023 [10] (see Figure 1.1).

As a result, the advertising industry has been testing and deploying alternatives to known third-party web tracking techniques, presenting new challenges to privacy-enhancing tools and transparency on the Internet. The main challenge in this context is that there are potentially many currently unknown techniques used by third parties to

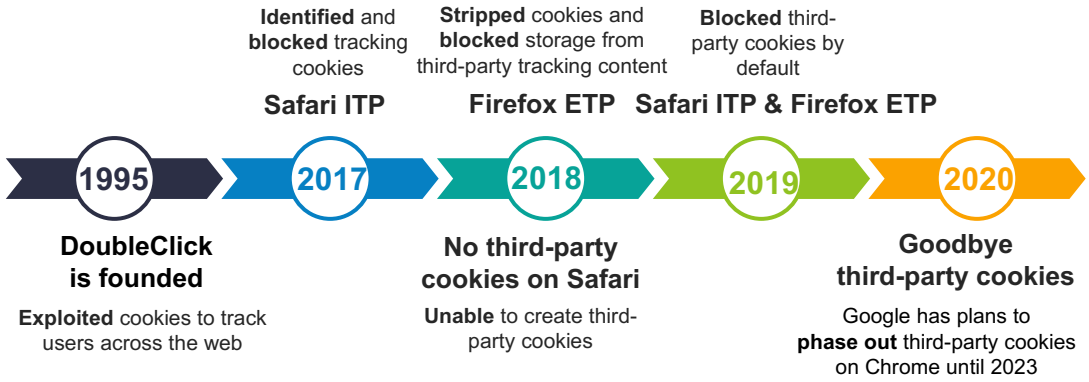


Figure 1.1: Brief history of third-party cookie.

circumvent existing countermeasures, especially approaches that leverage cooperation from the first-party to extract user data. In other words, the first-party has opened a window for third parties to track user activities. To overcome this problem, this dissertation aims at detecting, characterizing, then developing a countermeasure to protect user privacy against the first-party cooperation-based third-party web tracking.

1.2 Problem statements and research questions

The importance of online privacy measurement has long been recognized by the research community. The advertisement/tracking ecosystem has substantially grown in the last 10 years, building on standard and more advanced web tracking technologies. They are extremely useful for online advertisers and data brokers, however very frightful for the privacy of the users. In this dissertation, we intend to make the first attempt to elucidate the first-party cooperation-based third-party web tracking and develop the firm countermeasures to protect user privacy against these tracking techniques. In the light of the above discussion, our research questions will be broken down into four research questions (RQ):

1. RQ1: What are unknown third-party web tracking techniques based on first-party cooperation?

2. RQ2: How do they work and what are their impacts on the online ecosystem?

3. RQ3: Are current protection techniques effective against first-party cooperation-based third-party web tracking?

4. RQ4: How to protect the end-user from these tracking techniques?

1.3 Contributions

Our contribution, based on the four research questions above, follows two research axis, each one regarding a specific first-party cooperation-based third-party web tracking technique. The first part is a detection, characterization, and countermeasure of CNAME cloaking-based tracking. The second part is a detection, characterization, and countermeasure of PII leakage-based tracking.

1.3.1 CNAME cloaking-based tracking

Due to the increasing use of ad-blocking and third-party tracking protections, tracking providers introduced a new technique called CNAME cloaking. It misleads web browsers into believing that a request for a subdomain of the visited website originates from this particular website, while this subdomain uses a CNAME to resolve to a tracking-related third-party domain. This technique thus circumvents the third-party targeting privacy protections. For instance, website *example.com* embeds a first-party request by subdomain *a.example.com*, which points to a tracking provider *tracker.com* via a CNAME *x.tracker.com*. Because this request is in the first-party context, countermeasures that aim to block third-party tracking are effectively circumvented.

The goals of this part are to detect, characterize, and protect the end-user against CNAME cloaking-based tracking. Firstly, we characterize CNAME cloaking-based tracking by crawling the top pages of the Alexa Top 300,000 sites and analyzing the usage of CNAME cloaking with CNAME blocklist, including websites and tracking providers using this technique to track user's activities. We also point out that browsers and privacy protection extensions are largely ineffective to deal with CNAME cloaking-based tracking except for Firefox with a developer's version of the uBlock Origin extension. Secondly, we propose a supervised machine learning-based approach to detect CNAME cloaking-based tracking without the on-demand DNS lookup. We show that the proposed approach outperforms well-known tracking filter lists.

Finally, to circumvent the lack of DNS API in Chrome-based browsers, we design and implement a prototype of the supervised machine learning-based browser extension to detect and filter out CNAME cloaking tracking, called *CNAMETracking Uncloaker*. Our evaluation shows that *CNAMETracking Uncloaker* is able to filter out CNAME cloaking-based tracking requests without performance degradation when compared with the vanilla setting on the Chrome browser.

1.3.2 PII leakage-based tracking

Many popular websites give users the ability to sign up for their services, which requires personally identifiable information (PII). However, these websites embed third-party tracking and advertising resources, and as a consequence, the authentication flow can intentionally or unintentionally leak PII to these services. Since a user can be identified with PII, trackers can use it for tracking purposes, leading to further privacy leaks when cross-site, cross-browser, and cross-device tracking occurs. For instance, a first-party website, *site.com*, embeds a third-party tracking provider, *tracker.net*. After a user completes the sign-in flow on *site.com* by inputting his/her PII, a tracking script reads this PII and sends it to the tracking provider server, *tracker.net*. Because PII is a unique identifier, it allows *tracker.net* to match the user's browsing history across sites, browsers, and devices without using third-party cookies.

In this part, we document a PII-based tracking ecosystem that leverages user sign-up and sign-in flows (authentication flows) on first-party sites. To the best of our knowledge, this is the first in-depth analysis of PII leakage during authentication flows and the subsequent persistent web tracking mechanism that relies on this PII. By investigating the authentication flows for 307 popular shopping sites from the Tranco top 10,000 sites, we first discover that 42.3% of sites leak the PII to third-party services. Then, we present a previously unknown persistent web tracking technique based on PII leakage that enables tracking providers to generate and store a unique persistent identifier for a user with his/her browsing history on their tracking servers. We analyze 130 first-party senders along with 100 third-party receiver domains and show that PII leakage is a potentially important vector for online tracking for at least 20 providers. By measuring the presence of Online Behavioral Advertising (OBA), we confirm that the tracking providers use leaked PII in their advertising strategies for

cross-site, cross-browser, and cross-device targeting and personalization. In addition, we check the privacy policy of the 130 first-party senders and observe that they are not clear about PII exchange with third parties. Also, to provide a wider picture of current in-browser privacy protection techniques, we evaluate the effect of browsers and well-known blocklists against PII leakage. We point out that browsers are unable to deal with PII leakage except for Brave with its privacy-improving features, whereas blocklists reduce the number of leaked PII resources but do not completely fix this problem. Finally, we propose a hybrid approach to detect PII leakage by combining heuristic and supervised machine learning approaches. We show that the proposed approach outperforms well-known tracking filter lists.

1.4 Dissertation outline

The remaining of this dissertation are organized as follows:

- Chapter 2 provides the research background and related work. This chapter presents foundations of online privacy, third-party web tracking, privacy protection techniques, and related work to first-party cooperation-based third-party web tracking.
- Chapter 3 characterizes and defends the first-party cooperation-based third-party web tracking, namely CNAME cloaking-based tracking.
- Chapter 4 characterizes and defends the first-party cooperation-based third-party web tracking, namely PII leakage-based tracking.
- Chapter 5 discusses an in-depth exploration of the results, going into detail about the meaning of our findings about first-party cooperation-based third-party web tracking.
- Chapter 6 concludes this dissertation by summarizing our contributions and proposing several ideas to improve our research in the future.

2 Background and related works

In this chapter, we present different concepts and technologies used throughout this thesis in order to ease the understanding of the following chapters.

2.1 Online advertising ecosystem and online privacy

A major funding mechanism for the web is behavioral advertising - that is, advertising which is targeted based on a user's personal information, interests, and past behaviors, called Online Behavioral Advertising (OBA) [11–13]. It is a practice of tailoring advertising based on the tracking of user's online activities [14]. In order to achieve this target, tracking providers usually track an individual web usage history across multiple sites, browsers, and even devices. It requires a significant amount of data collection, ranging from data collected to better target an advertisement to the collection of purchase data to attribute the sale of a product to an advertiser.

Since the emergence of technology and the growth of the Internet, many studies have been conducted about user privacy concern [15–17], in which data collection

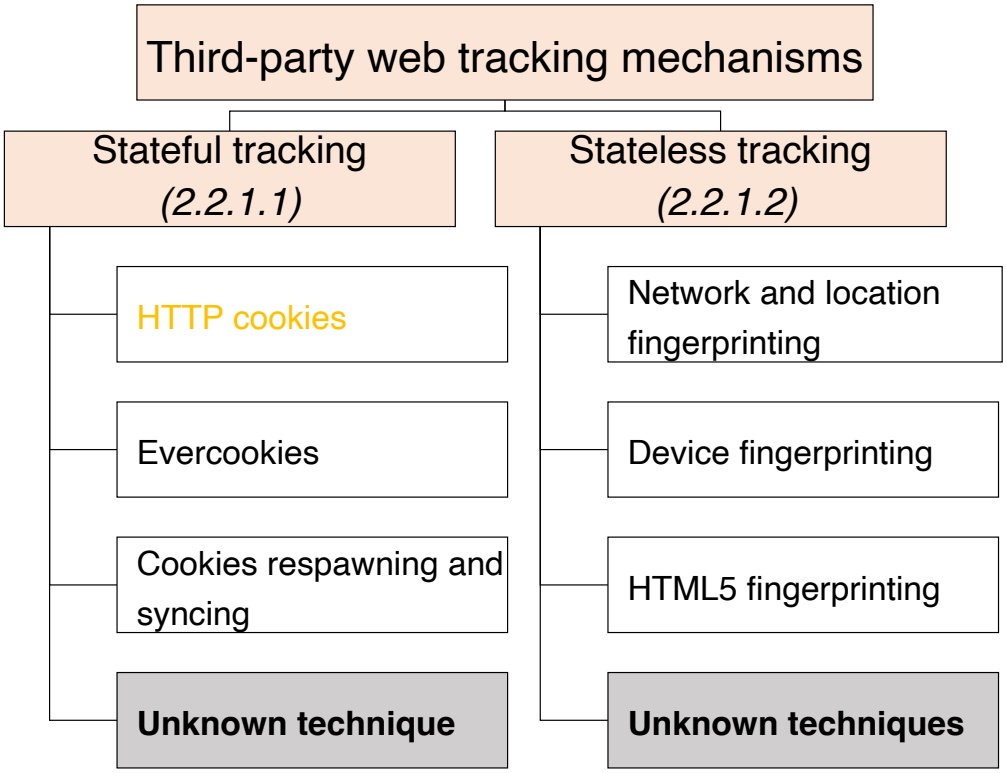


Figure 2.1: Summary of third-party web tracking mechanisms.

on individuals has become an increasingly common practice used by advertisement companies. Nearly every website a user visits records, aggregates, and shares information about that visit with third-party entities. The ubiquitous presence of the web in modern life is due in large part to its combination design - any website can pull in and make use of content from any number of entities. Web developers can take advantage of this to easily build and monetize web applications that provide a rich user experience. However, it has significantly reduced the cost of individual data collection and has greatly complicated efforts to protect user privacy [18]. Web tracking and individual data collection on the web are becoming a creepy and invasive idea [19, 20].

2.2 Third-party web tracking

Third-party web tracking refers to the practice of an entity, other than the domain directly visited by the user, which identifies and collects information about web users.

It is useful for a variety of purposes, such as online behavioral advertising that targets users with ads based on their profiles or interests. It may be considered as threatening for user privacy.

2.2.1 Third-party web tracking mechanisms

Many third-party web tracking techniques have been developed to maximize the benefits of tracing user browsing behavior. they are largely divided into two broad categories: stateful web tracking and stateless web tracking (see Figure 2.1).

2.2.1.1 Stateful web tracking

Stateful web tracking is a mechanism that recognizes users by retrieving information stored on user's devices. Specifically, third-party trackers can track users across websites by storing a unique identifier on the user's device. Modern web browsers provide several APIs that can be used to store this information, including HTTP cookies [21]; the evercookies -flash local shared objects (LSOs) [22], web storage API (i.e., localStorage and sessionStorage) [23], and the indexedDB API [24]; and cookies respawning and syncing [25, 26].

Cookie: It is the most currently known method to identify a user, which is a small piece of data placed in browser storage by the web server [21]. When a user visits a website for the first time, a cookie file with a unique user identifier (which could be randomly generated) is stored on the user's device. Then, the website can retrieve this identifier each time the user visits it unless the user deletes the cookie from his/her device. For instance, when a user accesses the website *site.com*, which loads a third-party resource *beacon.gif* from a third-party domain *tracker.net*. Along with this resource, the server *tracker.net* sets cookie header with a cookie, id=123. Such a cookie originates from *tracker.net* and is visible to this domain. From there, *tracker.net* gets the cookie id=123 along with it, helping it recognize the user and tracing all activities of this user on *site.com*. In the case this user visits another site, *new.com*, which also embeds a resource from *tracker.net*, the server at *tracker.net* will receive this cookie id and recognize the user, thus tracking this user across sites, *site.com* and *new.com* (see Figure 2.2).

Figure 2.2: Process flow of third-party cookie.

Evercookie: It is a process to store unique data on a user's device as an identification using multiple storage mechanisms, such as flash local shared objects (LSOs) [22], web storage API (i.e., localStorage and sessionStorage) [23], and the indexedDB API [24]. For instance, (1) when a user visits a site that includes requests content supported by Adobe Flash from *tracker.net*, (2) a text file flash cookies *xyz* that is sent by a web server *tracker.net* to a user's device;(3) From there, this flash cookie *xyz* is able to recognize returning users in this device (see Figure 2.3).

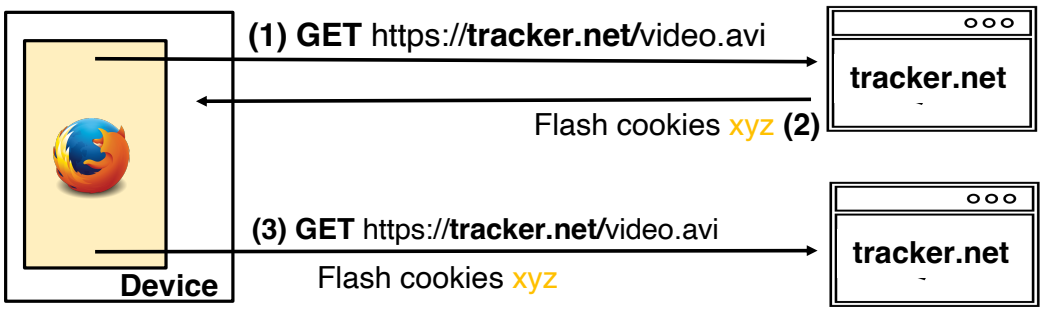
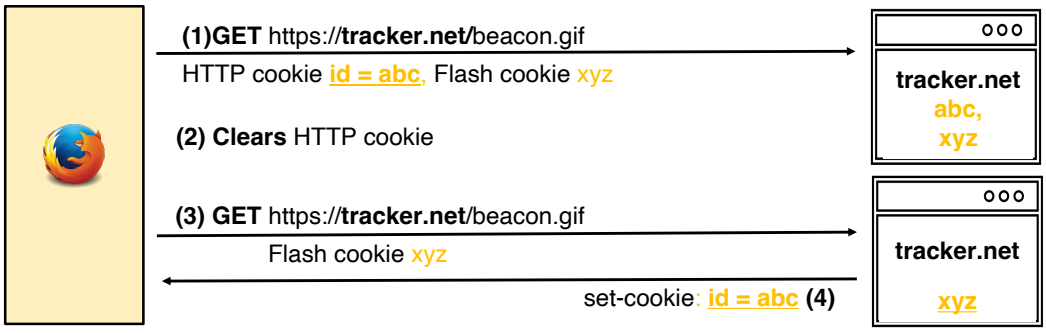
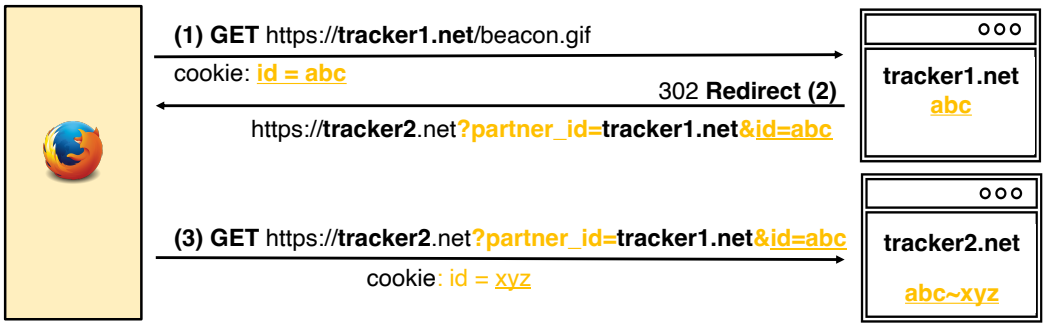


Figure 2.3: Process flow of evercookie - flash local shared objects.



(a) Cookie respawning.



(b) Cookie syncing.

Figure 2.4: Process flow of cookies respawning and cookies syncing

Cookies respawning: It is the process of recreating browser cookies from information that has been deleted. The third parties can take information stored in the evercookies - flash local shared objects and use it to recreate a cookie in a browser. For instance, when a user visits a site *site.com* that includes *tracker.net* as an embedded third-party tracker. (1) The browser makes a request to *tracker.net*, and included in this request is the tracking cookie, id=abc, and the flash cookie *xyz* set by *tracker.net*; (2) After a user clear cookie id=123 from the browser, (3) *tracker.net* retrieves its flash cookie xyz, (4 the server *tracker.net* uses the flash cookie to recreate a cookie, id=abc, stores in in the user browser (see Figure 2.4 (a)).

Cookies syncing: It is a process that enables the ad-tech partners to synchronize their cookies and share the incorporated user's data from different websites with each

other. For instance, when a user visits a site *site.com* that includes *tracker1.net* as an embedded third-party tracker. (1) The browser makes a request to *tracker1.net*, and included in this request is the tracking cookie set by *tracker1.net*; (2) *tracker1.net* retrieves its tracking ID from the cookie, and redirects the browser to *tracker2.net* , encoding the tracking ID into the URL; (3) The browser then makes a request to *tracker2.net* , which includes the full URL *tracker1.net* redirected to as well as *tracker2.net*'s tracking cookie; From there, *tracker2.net* can then link its ID for the user to *tracker1.net*'s ID for this user (see Figure 2.4 (b)).

2.2.1.2 Stateless web tracking

Stateless web tracking is a persistent tracking technique that does not require a tracker to set any state in the user's browser. Instead, trackers attempt to identify users by network and location fingerprinting, device fingerprinting, and HTML5 Fingerprinting. For instance, when a user visits a site *site.com* that includes *tracker.net* as an embedded third-party tracker, a piece of javascript, collects all relevant information about the user's device *xyz*. This information *xyz* is stored server-side *tracker.net*. When a user visits a site *site.com* again, or site *new.com*, *tracker.net* gets the fingerprinting *xyz* along with it, helping it recognize the user on server-side (see Figure 2.5).

Network and location fingerprinting: It is a process used to identify a device by the global network address and the IP-based geographical location of the user. By using network tools, the service is able to identify the name of the domain and the user's internet service provider as a factor for tracking user activities.

Device fingerprinting: It is a process used to identify a device by determining which technology, such as the operating system and browser plugins along with other active device settings, From there, it is able to recognize returning users by using this information.

HTML5 Fingerprinting: It is a process used to identify a device by using HTML5 API, such as canvas. Based on the fact that the same canvas image may be rendered differently on different computers, the canvas can be used as additional entropy in the web browser's fingerprinting and used for online tracking purposes.

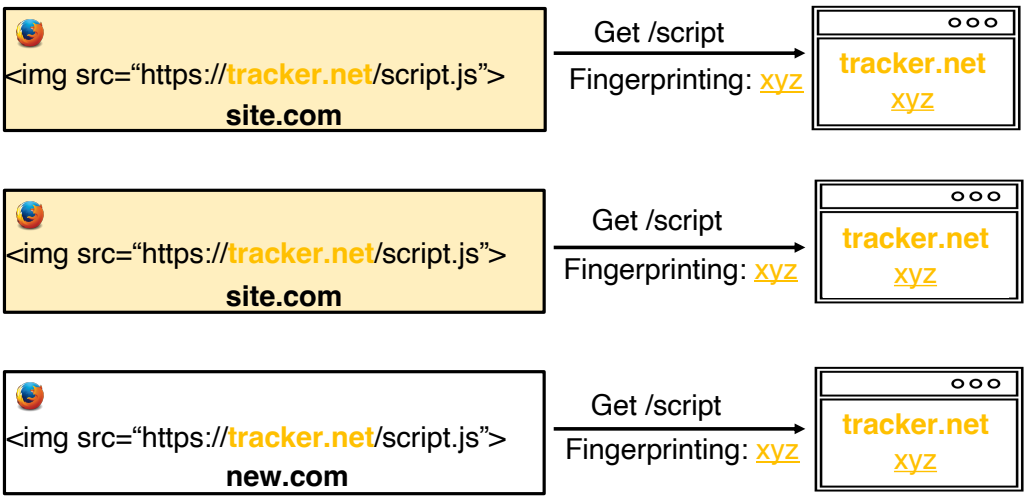


Figure 2.5: Process flow of stateless tracking.

2.2.2 Third-party web tracking measurement

Third-party web tracking techniques have been studied considerably in order to get a better understanding of the risks involved with these techniques. Many machine learning-based approaches have been proposed to detect and characterize third-party web tracking. Yamada et al. [27] analyzed traffic at the network gateway to monitor all tracking sites in the administrative network and constructs a graph between sites and their visited time to detect tracking sites. Metwalley et al. [28] developed an unsupervised detection method that inspects URL queries in HTTP(S) requests to detect tracking activities. To cut off the tracking chain of third-party web tracking, Pan et al. [29] developed TrackingFree which isolates unique identifiers into different browser principles so that the identifiers still exist but are not unique among different websites. Wu et al. [30] developed DMTrackerDetector which automatically detects third-party trackers offline to efficiently generate blocklists using structural hole theory and supervised machine learning. Ikram et al. [31] proposed one-class machine learning classifiers using syntactic and semantic features extracted from JavaScript programs to classify functional and tracking JavaScript programs. In addition, some approaches have detected the main mechanisms behind specific web tracking techniques.

Stateful web tracking measurement: Krishnamurthy and Wills [32] provide

much of the early insight into web tracking, showing the growth of the largest third-party organizations from 10% to 20-60% of top sites between October 2005 and September 2008 on 1200 popular websites. In the following years, studies show a continual increase in third-party tracking and in the diversity of tracking techniques [26,33–37]. Mayer et al. [33] surveyed the current policy debate surrounding third-party web tracking and explains the relevant technology and uses a fourth-party web measurement platform to collect HTTP requests, responses, and cookies. Roesner et al. [34] developed a client-side method for detecting and classifying five types of third-party trackers over 500 unique trackers on the 500 most popular and 500 less popular sites according to the Alexa ranking sites. Libert [35] provides a quantitative analysis of privacy-compromising mechanisms on one million popular websites. It shows that Google tracks users on nearly eight of ten sites in the Alexa top million sites. Schelter and Kunegis [36] performed a large-scale analysis of third-party trackers by extracting third-party embeddings from more than 41 million domains to study global online tracking. Papadopoulos et al. [26] design and implement a holistic mechanism to detect cookie syncing events in real-time using a year-long weblog from 850 real mobile users. Urban et al. [37] focus on the underlying information-sharing networks between online advertising companies in terms of client-side cookie syncing using graph analysis, which has negative effects on user privacy.

Stateless web tracking measurement: Several studies measured the prevalence of different fingerprinting mechanisms and evaluated existing countermeasures [18,38–42]. Eckersley [38] investigated the real-world effectiveness of browser fingerprinting algorithms by collecting these fingerprints from a sample of 470,161 browsers. Besson et al. [39] developed a generic framework for modeling hybrid monitors to evaluate the amount of information a web tracker learns by observing the output of a fingerprinting script for a particular browser configuration. Nikiforakis et al. [40] examine how web-based device fingerprinting currently works on the Internet by analyzing the code of three popular browser-fingerprinting code providers. Acar et al. [41] presented FPDetective, a fingerprinting detection framework that identifies web-based fingerprinters to perform a large-scale crawl of the Internet's most popular websites. They found that 404 sites in the top million deployed JavaScript-based fingerprinting and 145 sites of the top 10,000 sites leveraged Flash-based fingerprinting. Nikiforakis et al. [42] proposed a privacy mode present in modern browsers to combat device

fingerprinting by making subsequent visits to the same fingerprinter difficult to link together. Englehardt et al. [18] examine fingerprinting using several HTML5 APIs for fingerprinting on the top websites.

Overall, the privacy hazards of online web tracking have been studied extensively.

2.2.3 Third-party web tracking countermeasures

Several privacy protection techniques have been designed to protect end-users from third-party web tracking, including network-based blocking, extensions, and browser itself.

Network-based blocking methods use address-based blacklists in order to block access to certain domains (DNS blocking) and modify web traffic (interception proxies), which work independently of the underlying application or browser [43].

Some anti-tracking extensions work effectively to detect third-party web tracking, such as Ghostery [44], Disconnect [45], and uBlock Origin [46]. Some browsers also have built-in privacy protection features to protect end-users from third-party web tracking, such as Firefox [47], Brave [48], and Tor Browser [49]. Firefox introduces Enhanced Tracking Protection (ETP) feature from Firefox version 69. It blocks user profile from browsing behavior observation across websites [9]. Brave has a feature called Shields which protects user's privacy by blocking ads and trackers, cookies, malicious code, and malicious sites [48]. The Tor Browser is a browser based on the onion routing tool Tor and Mozilla's Extended Support Release (ESR) Firefox branch to enhance privacy and security. It includes both HTTPS-Everywhere and NoScript extensions which respectively enable HTTPS when possible, and allow users to block JavaScript [50].

2.3 First-party cooperation-based third-party web tracking

To bypass the countermeasures to protect user privacy from known third-party web tracking techniques, the advertisement/tracking ecosystem has built more advanced web tracking technologies. Here, we focus on first-party cooperation-based third-party web tracking techniques which are become a potential strategy for the online

advertising technology companies. These techniques can be classified into two fonds: CNAME cloaking-based tracking and PII leakage-based tracking. We present the state of the art of these concepts and technologies.

2.3.1 CNAME cloaking measurement and countermeasures

On the uBlock Origin's GitHub issues page, a user presented a website loading first-party request, which pointed to a tracking provider [51]. This issue was then addressed in several discussions [52] [53] [54].

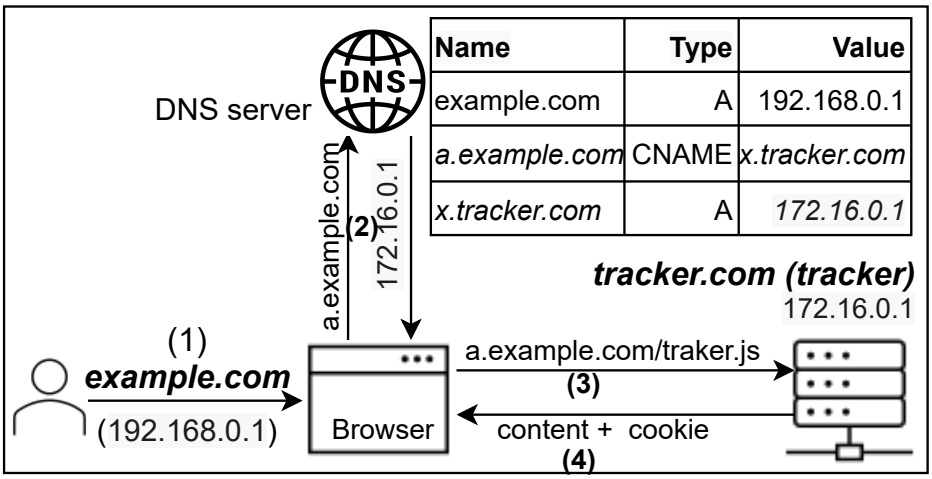


Figure 2.6: The process of the browser connecting to tracking provider by CNAME cloaking-based tracking

Figure 2.6 shows the process of the browser connecting to a third-party web tracking server by CNAME cloaking-based tracking to setup third-party cookies in the first-party context:

1. An end-user types the URL of website *example.com* (192.168.0.1) into his/her browser and presses return. This website embeds a subdomain *a.example.com*

2. The browser looks up *a.example.com* on the DNS server and finds an IP address 172.16.0.1 of tracking provider *tracker.com*.

3. Browser connects to the tracking provider web server *tracker.com* and asks for request script *tracker.js*.

4. Server *tracker.com* sends over the requested content along with cookies information. The browser accepts the content and these persistent cookies, which are stored under the domain name *example.com*. From there, the tracking provider *tracker.com* thus tracks the activities of this end-user on the website *example.com*.

CNAME cloaking-based tracking circumvents the third-party targeting privacy protections but it has a limitation. These persistent first-party subdomain-related cookies make it is more difficult for third parties to track users across websites by removing the simple mapping of each user to a single cookie linked to a single (third-party) domain.

Several countermeasures have been developed to protect end-user from CNAME cloaking-based tracking, including network-based DNS blocking and in-browser techniques. Network-based blocking methods are in use before web browsers support the conception of extensions [43]. NextDNS [52] and AdGuard [55] are applications at the DNS level, which require the wildcard match (domain and all its multi-level subdomains) against the domains in the CNAME cloaking blocklist. Nevertheless, NextDNS is a commercial product and it requires to install and configure the NextDNS client; AdGuard DNS is free in personal use, but end-users must set up their DNS servers and send their entire DNS traffic to the AdGuard server. In addition, Pi-hole [56] is a DNS sinkhole that protects end-user's devices from unwanted contents. However, the end-users have to install a supported operating system and Pi-hole on user's devices or separate hardware/appliance, then configure users router's DHCP options to force clients to use Pi-hole as their DNS server.

The in-browser privacy protection techniques not only improve user's privacy but can also increase user's browsing experience [57]. To make sure these potential advantages, some browser extensions also update themselves to block CNAME cloaking-based tracking resources. Adguard blocker [58], uBlocker Oringin [46], make a continuous effort to manually update first-party subdomains which are fronts for CNAME cloaking to these blocklists. It makes day-to-day filter lists updating tedious and time-consuming. As we evaluated in § 3.4, these extensions show the moderate detection performance to detect CNAME cloaking, except uBlock Origin with DNS API only supported by the Firefox browser. To keep up with this tracking technique, Safari and Brave add a new feature that keeps their users protected. The ITP Safari lowers the

duration of cookies set in the HTTP response created through JavaScript to defend with CNAME cloaking [59]. Meanwhile, the Brave embedded DNS resolver to block any request that has the canonical domain in their blacklist by default [60]. However, these browsers account for a small percentage of browser share [61].

To overcome these constraints, we propose an in-browser countermeasure that is based on a supervised machine learning-based method and a subdomain blocklist to detect CNAME cloaking-based tracking without the on-demand DNS lookup. To the best of our knowledge, we here propose the first in-browser extension that relies on machine learning techniques to protect the end-user from CNAME cloaking-based tracking (see Table 2.1).

Table 2.1: Detailed comparison of our countermeasure with other relevant works available in literature against CNAME cloaking.

Countermeasure	subdomain blocking list	DNS & blocking list	Machine learning	Cookie configuration	in-browser technique
NextDNS [52]		✓			
AdGuard DNS [55]		✓			
Pi-hole [56]		✓			
AdGuard blocker [58]	✓				✓
uBlock Origin (Firefox) [62]	✓	✓			✓
uBlock Origin [46]	✓				✓
Safari [63]				✓	✓
Brave [48]		✓			✓
Our work	✓		✓		✓

2.3.2 PII leakage measurement and countermeasure

PII is a piece of information that can be used to distinguish or trace an individual's identity either alone or when combined with other information that is associable to a specific individual [32].

PII leakage occurs when PII information leaks from a first party that gives it to a third-party. Recent work includes the detection of PII leakage to third parties in smartphone apps [64], data leakage due to browser extensions [65], and data leakage due to several web forms, including PII leakage from contact forms [66], mailing list subscription forms [67], registration forms [68], and authentication flows [69, 70]. Here,

we roughly follow the same methodology as in Refs. [68–70] to detect PII leakage to third parties, but we consider more detection methods (four and an additional combination with CNAME cloaking even when that information is obfuscated. We also present a tracking technique that relies on manipulating PII leakage. When a user's PII (i.e., email address, name, address) leaks to third-party domains (different from visited domains) to track user activities. This technique, called PII leakage-based tracking, uses PII to perform cross-site, cross-browser, and even cross-device user tracking.

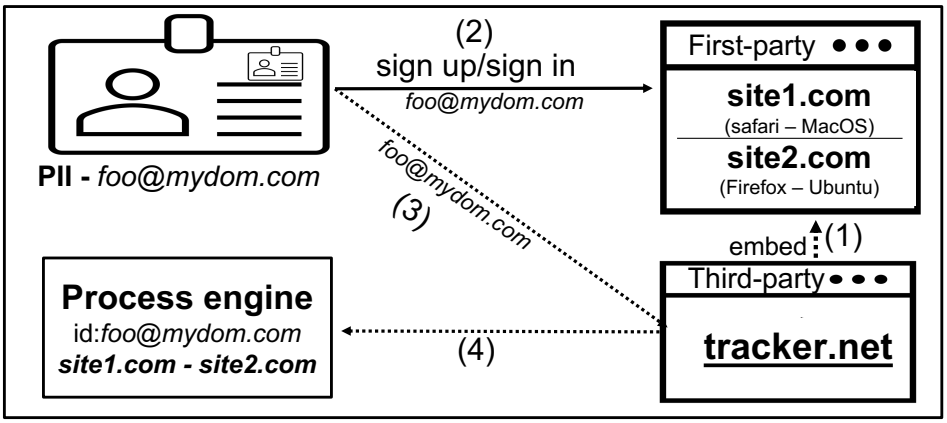


Figure 2.7: PII leakage-based tracking mechanism

Figure 2.7 shows the process of a third-party web tracking server tracking a user across sites, browsers, and devices

1. The first parties *site1.com* and *site2.com* embed the third-party resource *tracker.net* in their authentication flow.

2. An end-user access *site1.com*, *site2.com* and signup/signin that send his/her PII (email address *foo@mydom.com*) to these website server.

3. The PII (email address *foo@mydom.com*) is also sent to third-party server *tracker.net* during this authentication flow.

4. Server *tracker.net* uses the PII (email address *foo@mydom.com*) to track the activities of this end-user on the website *site1.com and site2.com* in different browsers and devices.

There are some existing methods to detect PII leakage resources partially. Some common browsers focus on Referrer-Policy by removing the PII that can be leaked to third parties via referer header [71, 72]. Starting from version 87, Firefox introduces a privacy-preserving default Referrer-Policy by trimming path and query string information from referer headers to prevent sites from accidentally leaking sensitive user data. Whereas Safari since version 13.0,4 downgrades all cross-site request referer headers to just the webpage's origin. Furthermore, Easylist [73], EasyPrivacy [74], and other filter lists can reduce the number of PII leakage by blocking requests; they however were not created for this purpose. Beyond these approaches, the Brave browser blocks tracking and advertising resources content from being loaded by default [75]. It also removes known tracking parameters from URLs and reduces the amount of information in the referer header to protect user privacy. However, this browser accounts for a small percentage of browser share [61, 76]. To overcome these limitations, we propose a countermeasure based on a hybrid method by combining heuristic and machine learning approaches to detect PII leakage that can easily be integrated into a browser and extension (see Table 2.2).

Table 2.2: Detailed comparison of our countermeasure with other relevant works available in literature against PII leakage.

Countermeasure	Referrer-Policy	Request blocking	Heuristic	Machine learning
Safari [72]	✓			
Firefox [71]	✓			
Brave [75]	✓	✓		
Blocklist [73, 74]		✓		
Our work			✓	✓

2.4 Summary

Third-party web tracking is a well-studied topic. However, existing work is insufficient to transparency the first-party cooperation-based third-party web tracking. It motivates us to perform this research to fulfill a better understanding of third-party web tracking. From there, improving and developing countermeasures to protect user privacy on the Internet against these tracking techniques, including CNAME cloaking-based tracking and PII leakage-based tracking.

3 CNAME cloaking-based tracking

3.1 Introduction

In the DNS hierarchy, a subdomain is any domain, which is underneath a main domain. Subdomains are used to organize or divide contents of a website into specific sections. For example, *a.example.com* and *b.example.com* are subdomains of domain *example.com*. The usage of DNS CNAME records coupled with Content Delivery Network (CDN) is increasingly commonplace to improve website load times, reduce bandwidth costs, and increase content availability and redundancy.

CNAME has also been used for user tracking. Tracking providers ask their clients to delegate a subdomain for data collection and tracking and link it to an external server using a CNAME DNS record [77]. This technique, called *CNAME cloaking-based tracking*, uses CNAME to disguise requests to a third-party tracker as first-party ones. We also define an HTTP request by this subdomain is a *request linked to CNAME cloaking-related tracking*.

There are some existing methods to detect CNAME cloaking-based tracking. Some

network-based blocking methods work at the DNS level, such as NextDNS [52], AdGuard DNS [55], Pi-hole [56] that use to get rid of online tracking. Furthermore, EasyPrivacy [78], AdGuard tracking protection [79], and other filter lists manually add new first-party subdomains which are fronts for CNAME cloaking to these blocklists. However, this approach will dramatically increase the size of the blocklists and these subdomains need to be updated frequently. Besides that, uBlock Origin since version 1.24.1b0 performs a DNS lookup of the hostname loading a resource to determine if the underlying subdomain is related to CNAME cloaking or not. Nevertheless, only Firefox allows uBlock Origin to block CNAME cloaking because the other browsers do not support DNS resolution API [46].

In this chapter, we provide a first in-depth analysis of CNAME cloaking-based tracking, propose a supervised machine learning-based method for the detection and implement *CNAMETracking Uncloaker* browser extension as a countermeasure. The main contributions of the paper are as follows. (1) We first characterize CNAME cloaking-based tracking in Alexa Top 300K sites. We detect 1,739 websites (0.58%) containing CNAME cloaking-based tracking in Alexa 300K sites as of January 2020 by matching with CNAME tracking filter lists (§ 3.5.1); Those websites are spread across many countries and categories. They use 24 tracking providers in total, and the most common one is Adobe (§ 3.3.4); By analyzing longitudinal snapshot crawled data of Alexa Top 100K sites (§ 3.3.5), we show that the usage of CNAME cloaking-based tracking steadily increases from 2016 to 2020; We then conduct further experiments to investigate the impact of giving consent to CNAME cloaking sites and confirm that there are no significant differences compared to the usage of this phenomenon before the user consent is obtained (§ 3.3.6). We also evaluate the detection ability of such tracking for major browsers and extensions (§ 3.4). (2) Next, we propose the supervised machine learning-based method to detect CNAME cloaking-based tracking without the on-demand DNS lookup (§ 3.5); Through the comprehensive analysis, we demonstrate the effectiveness of our method. (3) Finally, we design and implement a prototype browser extension of the supervised machine learning approach to protect the end-user against CNAME cloaking-based tracking, named *CNAMETracking Uncloaker* (§ 3.6). The current best countermeasure strongly depends on real-time name resolution (only supported by Firefox browser), but our extension intends to distinguish requests using CNAME cloaking-based tracking in Chrome-based browsers. Our experiment shows

Table 3.1: Summary of crawled data in Alexa Top 300K sites (Jan 2020).

Metrics			Numbers	Percentage
3rd party requests			14,640,568	54,27%
1st party requests	domain		5,919,965	21.94%
	subdomain	w/o CNAME	3,245,361	12.03%
		w/ CNAME	**3,172,304**	**11.76%**
Total requests			26,978,198	100%

that the performance overhead is acceptable when compared with the vanilla setting on the Chrome browser.

3.2 Data collection and blocklist-based detection

In this section, we describe the data collection and explain our methodology to detect CNAME cloaking-based tracking with blocklists.

3.2.1 Websites selection and data collection

The first step is the selection of websites that would be most appropriate for our work. We use the popularity index from Alexa [80] in all of our measurements, similar to past literature [43, 81, 82]. To characterize CNAME cloaking-based tracking, we use OpenWPM [82] to conduct large-scale automatic crawls on Alexa Top 300K sites. OpenWPM is based on Firefox version 52 and allows collecting all the HTTP/HTTPS requests emitted and their responses for each site. We performed the crawls with default settings in January 2020, with three IP addresses in Japan (Table 3.1).

In addition, to track the longitudinal behavior of CNAME cloaking-based tracking, we also rely on four other datasets (see Table 3.2). We collected two datasets on Alexa Top 100K sites with OpenWPM in April 2018 and January 2020. The other two datasets are publicly available in Princeton Web Census Data [82]. They were collected in January 2016 and February 2017 and targeted Alexa Top 100K sites. These datasets were also crawled with OpenWPM, so all the data sources are compatible and comparable. Note that the contents of Alexa lists are not the same among these four datasets because Alexa lists themselves are updated daily and change significantly

Table 3.2: Longitudinal snapshot datasets.

Time	Alexa	List gen.	Requests	Firefox version
Jan 2016	100K	01/2016	9,487,367	41
Feb 2017	100K	11/2016	10,964,374	45
Apr 2018	100K	03/2018	9,926,080	52
Jan 2020	100K	12/2019	9,647,506	52

from one day to the next [83]. The list used for each crawl is described in the "List gen." column of Table 3.2.

Furthermore, the instability of the Alexa Top list drastically increased in January 2018 [83]. Hence, to make a fair comparison, we also use the intersection (26,162 sites) of the four Alexa Top 100k sites above.

Note that, we found a publicly available HTTP Archive (HAR) dataset that provides historical data to quantitatively illustrate how the web is evolving [84]. However, we recognized two limitations of this dataset. First, The HAR dataset periodically crawled the top websites that come from the Chrome User Experience Report, but there is no ranking value in this dataset to assess whether end-users are actually impacted by CNAME cloaking. Second, there is no way to control its crawling and publishing schedule to obtain the up-to-date DNS data. Due to these reasons, we decided to use the dataset as described above for our measurement. We also put the analysis on this data set on Appendix A. We confirm the global consistency of our dataset with a publicly available HTTP Archive (HAR) dataset [84] (the details are shown in Table A.1).

3.2.2 Blocklist-based CNAME cloaking detection

3.2.2.1 CNAME lookup

First of all, we separate the generic Top-Level Domain (gTLD) and country-code top-level domain (ccTLD) from the visited website for all HTTP requests using the Public Suffix List [85]. We only keep subdomain of an HTTP request if it is not null and its second-level domain is the same as the visited website domain. We look up and check CNAME records for each subdomain. We then resolve each CNAME answer set

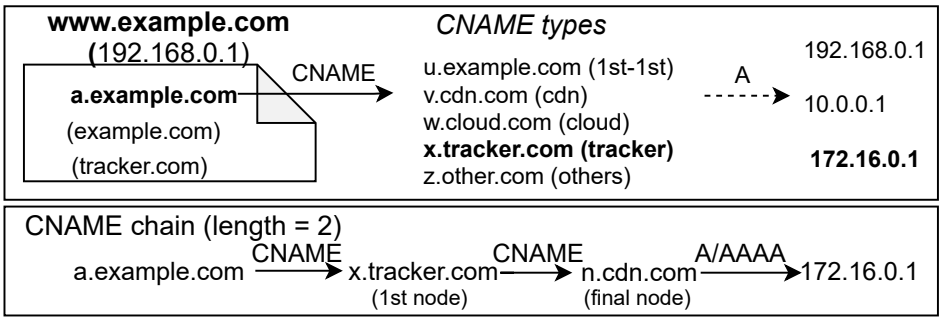


Figure 3.1: Overview of CNAME chain.

by DNS. We save all nodes in CNAME chain[1] (see Figure 3.1) to analyze the CNAME cloaking behind first-party requests.

We find that 45.73% of the HTTP requests are first-party requests in 2020 (Table 3.1). We then only keep 11.76% of the HTTP requests that contain first-party CNAME.

Looking up CNAMEs for the longitudinal data, we additionally check historical forward DNS (FDNS) datasets provided by Rapid7 [86]. The coverage of the FDNS data in our CNAME data is not perfect. It missed 10% of CNAMEs in 2018 and 30% in 2016 and 2017. We intend to use DNSDB [87] in future research to improve this coverage.

3.2.2.2 CNAME cloaking-based tracking detection with blocklists

To detect CNAME cloaking-based tracking, we use an approach based on wildcards matching of tracking blocklists.

First, we discard CNAME-related subdomains that are categorized as first-party types. We classify a CNAME chain as *first-party* if the domain of the final node in this

[1]CNAME chain corresponds to a series of CNAMEs from the initial first-party subdomain to all CNAME nodes before the resolution to an IP address (see Figure 3.1). We consider four CNAME types for a CNAME chain:

1. First-party type: The domain of the final node in a CNAME chain is the same as the domain of the considered HTTP request, or the IP addresses of both the final node and the second-level domain are the same (*u.example.com*).
2. CDN type: The domain of nodes in a CNAME chain is used for CDN service (*v.cdn.com*).
3. Cloud and other types: The domain of nodes in a CNAME chain is used for other activities, such as cloud storage or firewall (*w.cloud.com*, *z.other.com*).
4. Tracker type: The domain of nodes in a CNAME chain is used for tracking user activities (*x.tracker.com*).

chain is the same as the domain of the considered HTTP request, or if the IP addresses of both the final node and the second-level domain are the same.

We then intend to detect CNAME cloaking-based tracking inside the remaining subdomains. We apply wildcard matching based on well-known tracking blocklists: EasyPrivacy list [74] and AdGuard tracking protection filter [79]. EasyPrivacy list consists of nine sublists and the Adguard tracking filter list consists of eleven sublists. They contain many rules that remove all forms of tracking, including web bugs, tracking scripts, and information collectors, thereby protecting user personal data. Focusing on tracking domains, we select the *third-party tracking domains*, the *international third-party tracking domains*, the third-party domain from *third-party tracking services*, and the third-party domain from *International third-party tracking services* sublists from EasyPrivacy list and the *tracking servers list* sublist from AdGuard tracking protection filter as of February 5, 2020. These blocklists are partly overlapping. We build the union of the two blocklists above to make a *CNAME tracking filter list*. Then, we build regular expressions from tracking domains to match with CNAME behind all remaining subdomains. For example, *eulerian.net∧third-party* is changed to *.eulerian.net.$*. This rule matches any CNAME ending with *.eulerian.net.*; We can thus detect any CNAME cloaking-based tracking from tracking provider Eulerian [88]. Finally, we inspect individual CNAME nodes in all CNAME chains using this customized filter list. If any node in a CNAME chain is flagged by this list, we classify this CNAME chain as a potential *tracker* that flag by 62 domains from *our CNAME tracking filter list*.

To avoid false positives, we then group these CNAME chains by domain and inspect them manually one by one. We first validate them by observing the activities which store an uniquely cookie in the browser under the visited domain name. We also gather information about these domains to identify whether they belong to any tracking provider. Using this analysis, we finally consider 28 domains are used for CNAME cloaking-based tracking and flag these chains as *tracker*.

We furthermore use CDN lists [89, 90] to check if remaining CNAME chains are *CDN*. If it is not the case, we consider them as *Others*.

3.3 Characterizing CNAME cloaking-based tracking

3.3.1 CNAME cloaking-based tracking analysis

Having gathered the CNAME chains using CNAME cloaking-based tracking, we concentrate on analyzing websites and tracking providers linked to CNAME cloaking-based tracking.

We consider the ranking, the country, and the category of websites containing CNAME cloaking-based tracking. For the website ranking, we assess how a real user would be affected in the real world by this type of tracking by examining the Empirical Cumulative Distribution Function (ECDF) of these websites. For the website country using CNAME cloaking-based tracking, we analyze them based on the top-level domain (TLD), Whois information, and IP Geolocation. First of all, if the TLD of a website corresponds to a country (i.e., ccTLD), we attribute that website to the country. By doing that, we identify the country of 94,560 websites. Then, for international TLDs, we use Whois information to determine 171,370 websites' country. Finally, for 34,070 remaining domains, we use the IP Geolocation to determine the country. We are aware that, if a website uses cloud-based security, proxy, or DNS-based service, then the geolocation of returned IP address could be unreliable for our purpose. However, this error was negligible, especially, there are a small number of such CNAME cloaking websites as shown in a later section (see § 3.3.3). In addition, IP Geolocation sometimes returns incorrect results [91]. To overcome this limitation, we make a majority voting via ip-api.com [92], freegeoip.app [93], and MaxMind [94] to give more robustness to the Geolocation assignment. In the 1,307 cases of three databases that return different results or return null, we set these websites to an unknown country. For the website category, we use FortiGuard Web Filtering [95] dataset from January 2020 for the website category classification.

Finally, we consider tracking providers behind CNAME cloaking-based tracking by linking 28 domains are used for CNAME cloaking to 24 tracking providers using Disconnect's blocklist [96].

Table 3.3: CNAME types of first-party request by subdomain (Alexa 300K sites in 2020).

Metric	1st-1st	Tracker	CDN, Cloud and others
HTTP requests	1,839,728/57.99%	**3,484/0.11%**	1,329,092/41.90%
Subdomains	48,365/39.47%	**1,803/1.47%**	72,376/59.06%

3.3.2 CNAME chains structure

In this section, we focus on the characteristics of CNAME chains for the first-party subdomain in Alexa Top 300K sites. Firstly, we present the CNAME usage of first-party requests by subdomain in Table 3.3. The most common CNAME type is requests referring to resources of the first party (57.99%). CDN and cloud also represent a large proportion of CNAME types (41.90%). Overall, we detect 3,484 CNAME cloaking-based tracking URLs. Furthermore, we find that these URLs belong to 1,739 websites (0.58%) on Alexa Top 300K sites.

Then, we breakdown the number of nodes in CNAME chains for first-party subdomains in our latest dataset (Alexa Top 300K sites in 2020) in Figure 3.2. We observe that about 80% of CNAME chains are very simple, just consisting of one CNAME. However, we also observe longer chains whose maximum length is six. These longest chains are mainly used by Microsoft likely for load balancing. This result suggests that checking only the first CNAME might be not enough for detecting CNAME cloaking-based tracking, because tracker websites may appear in intermediate nodes in the chain.

Finally, we show the breakdown of CNAME types regarding their position in CNAME chains in Figure 3.3. We note that the position represents the location of a CNAME in a CNAME chain. For example, the first position of a CNAME chain with two nodes *x.tracker.com* and *n.cdn.com* is the CNAME *x.tracker.com.*. In Alexa Top 300K sites, the tracking-related domain inside a CNAME chain is mainly located at the first position. We however also observe some tracking domains in the second position.

3.3.3 Websites using CNAME cloaking-based tracking

Next, we focus on the characteristics of websites containing CNAME cloaking-based tracking.

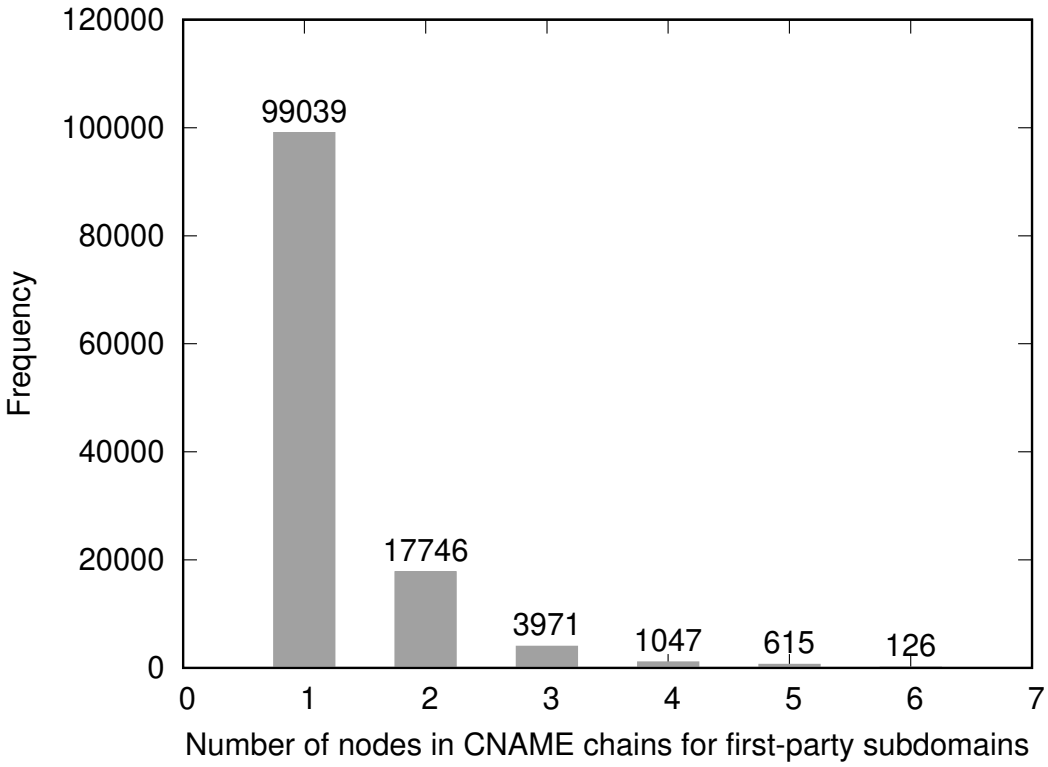


Figure 3.2: The number of nodes in CNAME chains for first-party subdomains (Alexa Top 300K sites in 2020).

Figure 3.4 presents the Empirical Cumulative Distribution Function (ECDF) of the Alexa ranking of websites containing CNAME cloaking-based tracking. These websites are spread across the Alexa ranking. It illustrates that 30% of the CNAME cloaking-based tracking belongs to the top 20K websites. Popular websites use more CNAME cloaking-based tracking.

Then, we discuss the website category of websites containing CNAME cloaking-based tracking that shown in Figure 3.5. For 1,739 websites containing CNAME cloaking, the percentages of websites in Business, Information Technology, Shopping and Finance are 22.0%, 17.3%, 11.8%, and 9.7%, respectively. In addition, for the proportion of website using CNAME cloaking inside each category, the percentages of these websites account for 0.6%, 0.6%, 1.2%, and 2.4%, respectively. Overall, various website categories use CNAME cloaking.

Next, we analyze the website country of websites containing CNAME cloaking-

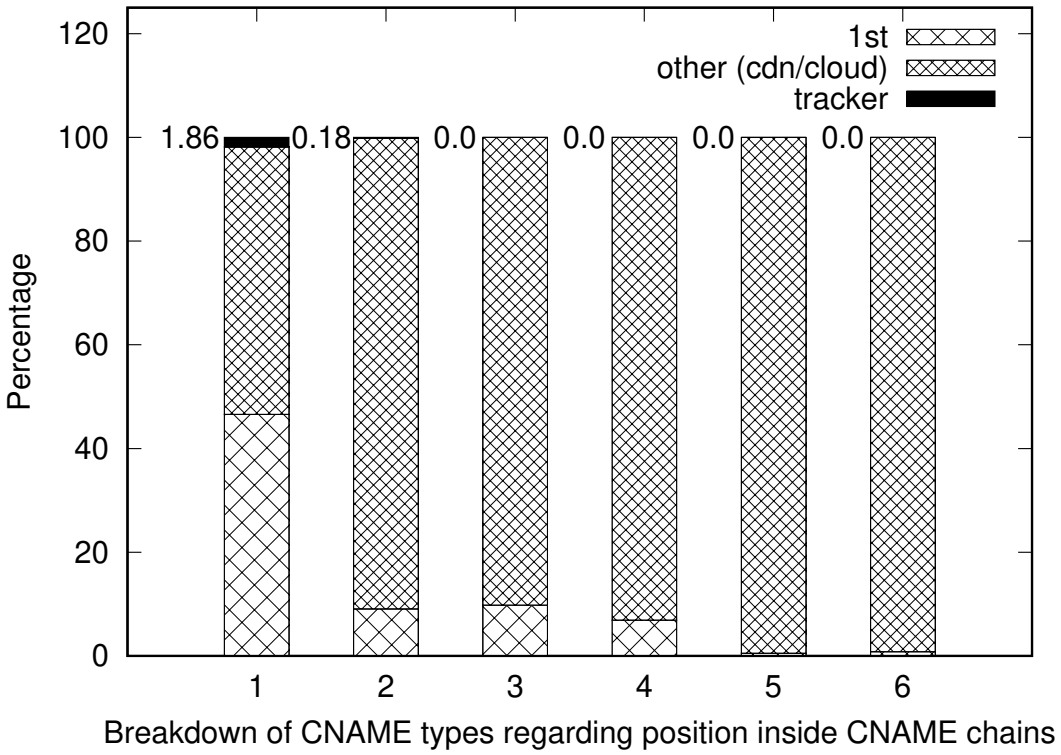


Figure 3.3: Breakdown of CNAME types regarding position inside CNAME chains (Alexa Top 300K sites in 2020).

based tracking[2] that shown in Figure 3.6. We observe that 55.1% of websites are located in the United States, 5.6% are located in Germany, 5.3% are located in the United Kingdom, 5.0% are located in Japan, 4.9% are located in Canada, and other countries have significantly lower percentages. In addition, for the proportion of websites using CNAME cloaking inside each country, the percentage of the United States, Germany, the United Kingdom, Japan, and Canada are 1.1%, 1.7%, 1.2%, 1.0%, and 0.7%, respectively. Overall, there is not a big difference among websites using CNAME cloaking regarding country.

In summary, we intended to investigate any biases, but we do not observe significant

[2]The website country for 1,739 sites containing CNAME cloaking-based tracking is determined by ccTLD (434 sites; 24.96%), Whois (1,054 sites; 60.61%), and IP Geolocation (251 sites; 14.43%). The websites detected by the IP Geolocation are identified as the United States and Canada (222 sites), European countries (22 sites), and others (seven sites). We manually confirm that most results are not affected by CDN.

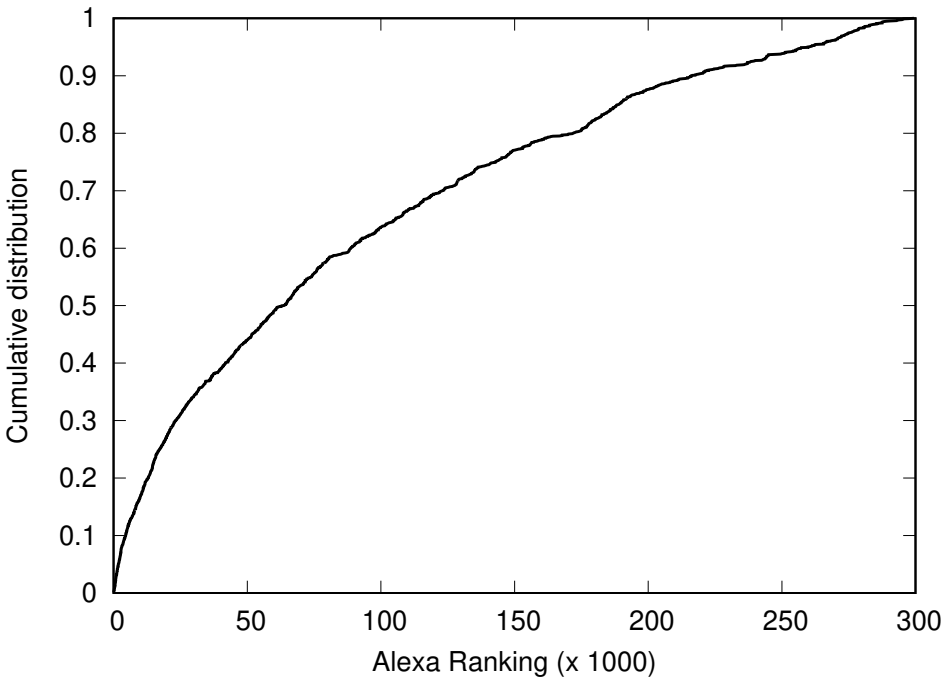


Figure 3.4: ECDF of the Alexa ranking of websites containing CNAME cloaking-based tracking (Alexa Top 300K sites in 2020).

biases regarding website categories and website countries of sites containing CNAME cloaking-based tracking. In contrast, websites using this tracking technique are widely spread in many countries and categories.

3.3.4 Tracking providers using CNAME cloaking-based tracking

We provide the breakdown of tracking providers behind CNAME cloaking-based tracking in Figure 3.7. We confirm 24 tracking providers using this technique. The major player in Alexa Top 300K sites is Adobe (52.5%). Besides Adobe, we see some well-known tracking providers, such as Pardot [97], Act-on [98], Oracle [99], and Webtrekk [100] (25.7%, 6.3%, 3.0%, and 2.5%, respectively).

Moreover, Table 3.4 shows the breakdown of tracking providers inclusion in website by website category. We observe that tracking providers were distributed in different types of websites, except Intent (Travel category with 92%). In addition, Adobe and Pardot are the most popular tracking providers in almost all categories. Furthermore, Table 3.5 shows the breakdown of the tracking providers inclusion in

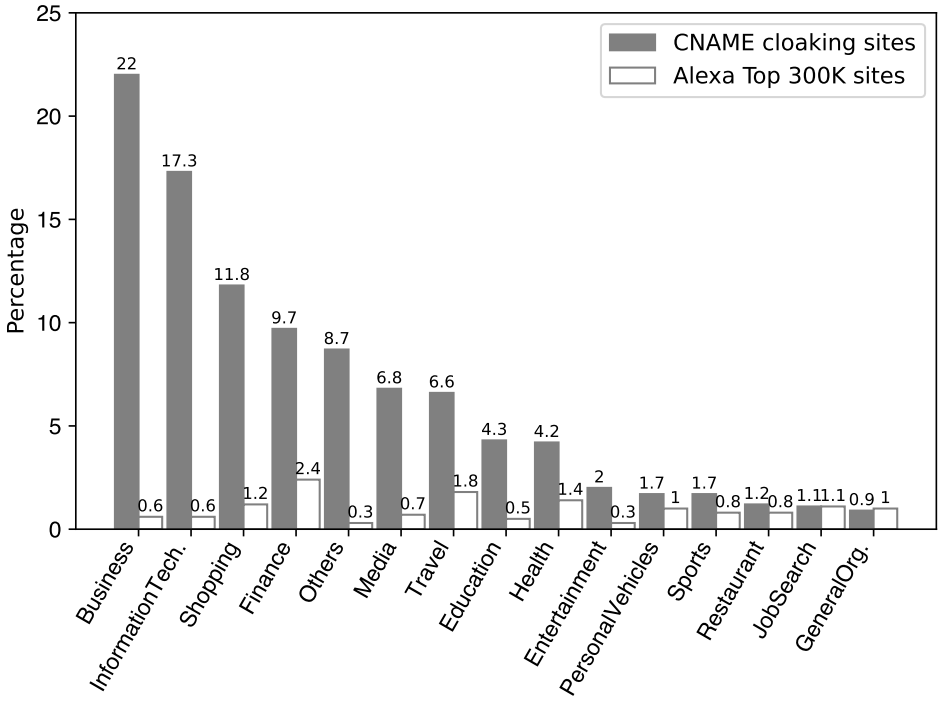


Figure 3.5: Breakdown of websites containing CNAME cloaking-based tracking by website category (Alexa Top 300K sites in 2020).

Table 3.4: Breakdown of tracking providers inclusion in website by website category. The values have the following meaning: raw/percentage for category/percentage for tracking provider. The significant percentages (>20%) are shown in bold.

Category	Adobe	Pardot	Act-On	Oracle	Webtrekk	Eulerian	Segment	Intent	PostafiliatePro	Others	Total
Business	141/15.1/**36.5**	157/**34.5**/**40.7**	39/**35.1**/10.1	13/**24.5**/3.4	14/**31.1**/3.6	5/11.9/1.3	7/**24.1**/1.8	0/0/0	2/16.7/0.5	8/11.6/2.1	**386**/NA/100
Information Technology	95/10.2/**30.4**	139/**30.5**/**44.6**	25/**22.5**/8.0	16/**30.2**/5.1	6/13.3/1.9	7/16.7/2.2	8/**27.6**/2.6	0/0/0	4/**33.3**/1.3	12/17.4/3.8	**312**/NA/100
Shopping	157/16.9/**73.0**	5/1.1/2.3	1/0.9/0.5	3/5.7/1.4	10/**22.2**/4.7	10/**23.8**/4.7	4/13.8/1.9	0/0/0	2/16.7/0.9	23/**33.3**/10.7	**215**/NA/100
Finance	117/12.6/**68.8**	24/5.3/14.1	7/6.3/4.1	6/11.3/3.5	4/8.9/2.4	3/7.1/1.8	1/3.4/0.6	1/4.0/0.6	0/0/0	7/10.1/4.1	**170**/NA/100
Media	96/10.3/**80.0**	5/1.1/4.2	2/1.8/1.7	0/0/0	5/11.1/4.2	3/7.1/2.5	1/3.4/0.8	1/4.0/0.8	0/0/0	7/10.1/5.8	**120**/NA/100
Travel	64/6.9/**54.2**	10/2.2/8.5	6/5.4/5.1	0/0/0	2/4.4/1.7	8/19.0/6.8	0/0/0	23/**92.0**/19.5	0/0/0	5/7.2/4.2	**118**/NA/100
Education	18/1.9/**24.0**	41/9.0/**54.7**	8/7.2/10.7	4/7.5/5.3	0/0/0	0/0/0	4/13.8/5.3	0/0/0	0/0/0	0/0/0	**75**/NA/100
Health	45/4.8/**61.6**	19/4.2/**26**	3/2.7/4.1	2/3.8/2.7	1/2.2/1.4	0/0/0	1/3.4/1.4	0/0/0	2/16.7/2.7	0/0/0	**73**/NA/100
Entertainment	31/3.3/**88.6**	1/0.2/2.9	2/1.8/5.7	0/0/0	0/0/0	1/2.4/2.9	0/0/0	0/0/0	0/0/0	0/0/0	**35**/NA/100
Personal Vehicles	24/2.6/**80.0**	2/0.4/6.7	0/0/0	2/3.8/6.7	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0	2/2.9/6.7	**30**/NA/100
Sports	20/2.1/**69.0**	4/0.9/13.8	1/0.9/3.4	1/1.9/3.4	0/0/0	1/2.4/3.4	2/6.9/6.9	0/0/0	0/0/0	0/0/0	**29**/NA/100
Restaurant	16/1.7/**80.0**	3/0.7/15.0	0/0/0	0/0/0	0/0/0	1/2.4/5.0	0/0/0	0/0/0	0/0/0	0/0/0	**20**/NA/100
Job Search	10/1.1/**52.6**	7/1.5/**36.8**	2/1.8/10.5	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0	**19**/NA/100
General Organizations	6/0.6/**35.3**	7/1.5/**41.2**	2/1.8/11.8	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0	2/2.9/11.8	**17**/NA/100
Others	91/9.8/**59.5**	31/6.8/**20.3**	13/11.7/8.5	6/11.3/3.9	3/6.7/2.0	3/7.1/2.0	1/3.4/0.7	0/0/0	2/16.7/1.3	3/4.3/2.0	**153**/NA/100
Total	**931**/100/NA	**455**/100/NA	**111**/100/NA	**53**/100/NA	**45**/100/NA	**42**/100/NA	**29**/100/NA	**25**/100/NA	**12**/100/NA	**69**/100/NA	**1,772**/NA/NA

website by website country. Tracking providers cooperating with websites such as Act-on, PostafiliatePro, Pardot, Adobe, and Oracle are mainly located in the United States (80.2%, 66.7%,61.5%, 56.7%, and 52.8%, respectively). We also observe that some tracking providers are mainly located in specific countries, e.g., Eulerian in France

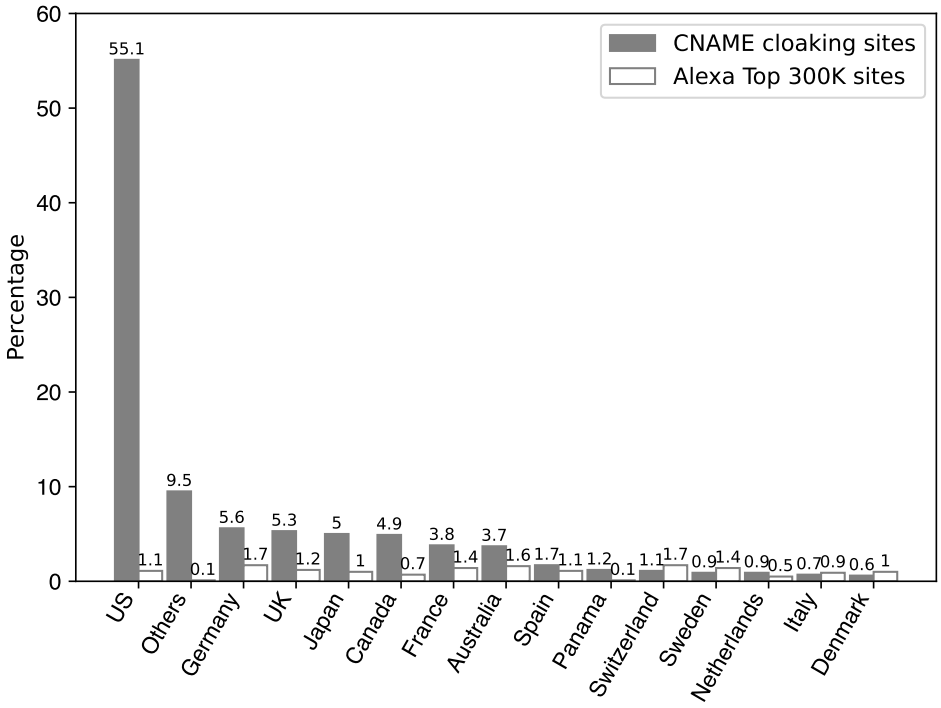


Figure 3.6: Breakdown of websites containing CNAME cloaking-based tracking website country (Alexa Top 300K sites in 2020).

Table 3.5: Breakdown of tracking providers inclusion in website by website country. The values have the following meaning: raw/percentage by country/percentage by tracking provider. The significant percentages (>20%) are shown in bold.

Country	Adobe	Pardot	Act-On	Oracle	Webtrekk	Eulerian	Segment	Intent	PostafiliatePro	Others	Total
United States	528/**56.7**/**54.5**	280/**61.5**/**28.9**	89/**80.2**/9.2	28/**52.8**/2.9	1/2.2/0.1	1/2.4/0.1	11/**37.9**/1.1	5/**20.0**/0.5	8/**66.7**/0.8	18/**26.1**/1.9	**969**/NA/100
Germany	38/4.1/**35.5**	8/1.8/7.5	1/0.9/0.9	1/1.9/0.9	31/**68.9**/**29.0**	1/2.4/0.9	0/0/0	2/8.0/1.9	0/0/0	25/**36.2**/**23.4**	**107**/NA/100
United Kingdom	58/6.2/**62.4**	22/4.8/**23.7**	2/1.8/2.2	2/3.8/2.2	1/2.2/1.1	2/4.8/2.2	1/3.4/1.1	3/12.0/3.2	0/0/0	2/2.9/2.2	**93**/NA/100
Japan	36/3.9/**40.9**	51/11.2/**58.0**	0/0/0	1/1.9/1.1	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0	**88**/NA/100
Canada	50/5.4/**58.1**	22/4.8/**25.6**	4/3.6/4.7	3/5.7/3.5	0/0/0	6/14.3/7	0/0/0	1/4.0/1.2	0/0/0	0/0/0	**86**/NA/100
France	15/1.6/**21.4**	14/3.1/**20.0**	1/0.9/1.4	2/3.8/2.9	0/0/0	23/**54.8**/**32.9**	0/0/0	3/12.0/4.3	0/0/0	12/17.4/17.1	**70**/NA/100
Australia	48/5.2/**71.6**	10/2.2/14.9	2/1.8/3.0	3/5.7/4.5	0/0/0	0/0/0	4/13.8/6.0	0/0/0	0/0/0	0/0/0	**67**/NA/100
Spain	20/2.1/**64.5**	0/0/0	0/0/0	0/0/0	1/2.2/3.2	8/19.0/**25.8**	0/0/0	0/0/0	0/0/0	2/2.9/6.5	**31**/NA/100
Panama	2/0.2/10.0	10/2.2/**50.0**	1/0.9/5.0	2/3.8/10.0	0/0/0	0/0/0	4/13.8/**20.0**	0/0/0	1/8.3/5.0	0/0/0	**20**/NA/100
Switzerland	11/1.2/**57.9**	5/1.1/**26.3**	1/0.9/5.3	1/1.9/5.3	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0	1/1.4/5.3	**19**/NA/100
Sweden	7/0.8/**43.8**	6/1.3/**37.5**	2/1.8/12.5	1/1.9/6.3	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0	**16**/NA/100
Netherlands	9/1.0/**60.0**	3/0.7/**20.0**	0/0/0	0/0/0	2/4.4/13.3	0/0/0	0/0/0	0/0/0	0/0/0	1/1.4/6.7	**15**/NA/100
Italy	8/0.9/**61.5**	1/0.2/7.7	0/0/0	1/1.9/7.7	2/4.4/15.4	0/0/0	0/0/0	0/0/0	0/0/0	1/1.4/7.7	**13**/NA/100
Denmark	8/0.9/**80.0**	0/0/0	0/0/0	2/3.8/**20.0**	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0	**10**/NA/100
Others	93/10.0/**55.4**	23/5.1/13.7	8/7.2/4.8	6/11.3/3.6	7/15.6/4.2	1/2.4/0.6	9/**31.0**/5.4	11/**44.0**/6.5	3/**25.0**/1.8	7/10.1/4.2	**168**/NA/100
Total	**931**/100/NA	**455**/100/NA	**111**/100/NA	**53**/100/NA	**45**/100/NA	**42**/100/NA	**29**/100/NA	**25**/100/NA	**12**/100/NA	**69**/100/NA	**1,772**/NA/NA

(54.8%) and Webtrekk in Germany (68.9%). Again, Adobe and Pardot are the most popular tracking providers in almost all countries, except France (Eulerian with 32.9%).

Finally, we further investigate the number of tracking providers on each website. Most websites (1,707) deploy only one tracking provider, as expected. However, we

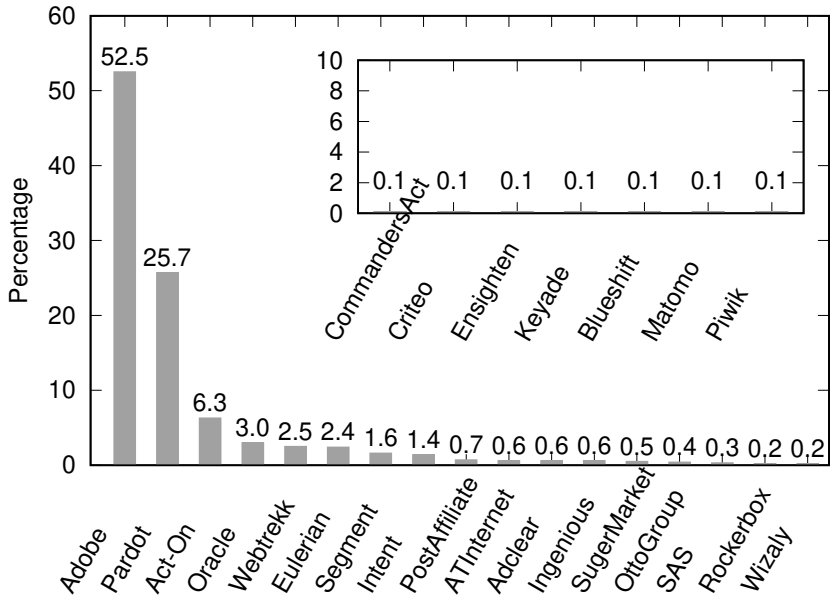


Figure 3.7: Tracking providers providing CNAME cloaking-based tracking (Alexa Top 300K sites in 2020).

also find 31 websites using two providers, and one website *mytoys.de* using three providers (Webtrekk, Otto Group, and Adclear). Typical pairs of the providers are the combination between Adobe and other tracking providers, such as (Adobe and Oracle), (Adobe and Webtrekk), or (Adobe and Pardot). We do not identify any plausible reasons of deploying multiple providers, but they might be used for different purposes (e.g., analytics and advertisement).

We conclude that, besides the biggest player Adobe, CNAME cloaking tracking providers operate on many website categories and countries.

3.3.5 Longitudinal analysis of CNAME cloaking-based tracking

In this section, we analyze the longitudinal evolution of the number of websites using CNAME cloaking-based tracking. Figure 3.8 indicates the number of websites using CNAME cloaking-based tracking in Alexa 100K sites. We combine two crawled datasets and two DNS lookup datasets: (1) for the crawled data, the number of websites in each Alexa 100K and those in the overlap among all Alexa 100K datasets (26,126 sites); (2) for two DNS lookup datasets, DNS lookup in 2020 and lookup with the FDNS

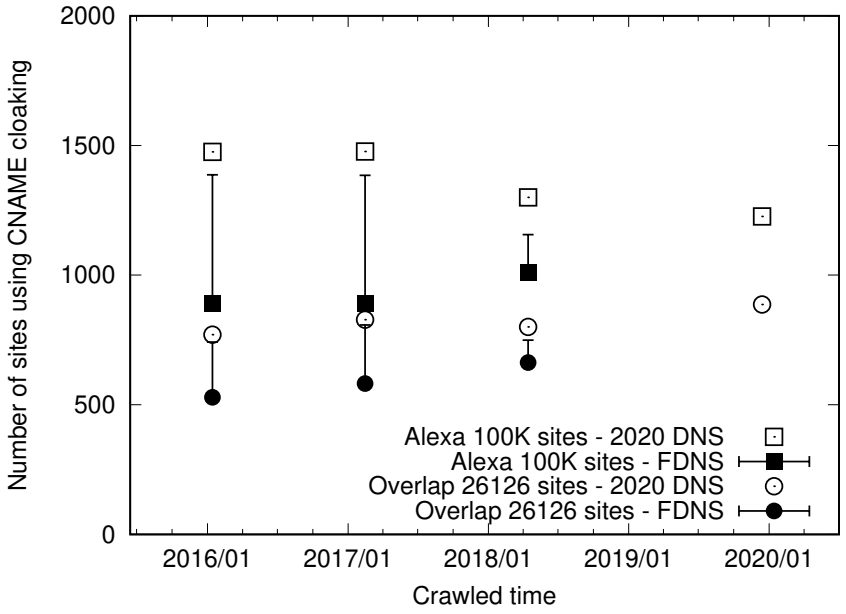


Figure 3.8: Websites containing CNAME cloaking-based tracking along time.

data (collected in February 2017, the oldest available snapshot, and June 2018). We then plot four combinations: the number of websites in each Alexa 100K sites with 2020 DNS (white rectangles) and with FDNS (black rectangles). Those in the overlap among all Alexa 100K datasets with DNS in 2020 (white circles) and with FDNS (black circles). The error bars in the figure show the number of unsolved CNAMEs due to the coverage of the FDNS data.

We discuss the growth of websites introducing CNAME cloaking-based tracking over the years. At a glance, the number of websites containing CNAME cloaking-based tracking is slightly decreasing in Alexa Top 100K websites with the latest DNS (white rectangles). However, this decrease is due to biases of DNS lookup. Considering the historical DNS data (black rectangles), we conjecture the presence of an increasing trend in the use of CNAME cloaking. However, the large number of unsolved CNAMES in 2016 and 2017 (represented by the error bars in the figure) does not allow to confirm this. We see an increasing trend in the overlapping websites (white and black circles) with smaller error bars. Although the unsolved CNAMES in 2016 and 2017 for the yearly Alexa limit the strength of our conclusion, the evolution between 2018 and 2020 for yearly Alexa, and the overall trend in the overlapping websites, allow us to confirm

an increasing trend along the observed years.

3.3.6 Impact of giving consent to CNAME cloaking sites

We also conduct extra experiments to evaluate the impact of providing consent as legally required by General Data Protection Regulation (GDPR) [6] to websites with CNAME cloaking. We pick up 1000 sites in Alexa top 300K sites (top 500 sites, randomly middle 250 sites, and randomly bottom 250 sites). On a clean browser session, we load the website. If there is no cookie notification or only a text simply informing the users about the site's usage of cookies, we stop there. We find 917 sites (91.7%) as *no banner* (the publishers do not inform the end-user of data collection) and 19 sites (0.19%) as *notification only* (the notification simply informing the users about the site's usage of cookies) category. For 64 (0.64%) remaining sites, we crawl them twice. In the first time, we save all requests and responses in these sites without human manipulation and find 31 sites (0.31%) as *Accepted only* (the notification does not offer a way to refuse consent) and 33 sites (0.33%) as *More options* (the user can make their choice in the cookies notification by clicking *accept*, *reject*, or *more setting*) category. In the second time, we click to accept consent on the banner, record the requests (if any). Comparing the difference between *Accepted only* and *More options* categories in the two crawls, we confirm that four websites already embed a subdomain-related request to hidden CNAME cloaking-based tracking without obtaining user consent. These results demonstrate that there is no significant effect by giving consent to the CNAME cloaking sites in our measurement.

3.4 Measuring the effectiveness of the current in-browser protection techniques against CNAME cloaking

We analyze and compare browsers and extensions regarding privacy protection against CNAME cloaking-based tracking.

Table 3.6: Detection performance: EasyPrivacy list and AdGuard tracking protection filter (Alexa Top 300K sites in 2020).

Metric	AdGuard Tracking	EasyPrivacy	All (combined)
HTTP requests	1,433/41.13%	2,707/77.70%	2,713/77.87%
Subdomains	444/24.63%	1,313/72.82%	1,316/72.99%
Sites	422/24.27%	1,262/72.57%	1,265/72.74%

3.4.1 Blocklists

In order to block CNAME cloaking-based tracking, EasyPrivacy [74] and AdGuard tracking protection [79] require the identification of first-party subdomains which are fronts for CNAME cloaking. They follow the Adblock Plus filter syntax. For example, EasyPrivacy has a rule to block tracking provider Eulerian: *f7ds.liberation.fr^*. Thus, when website *liberation.fr* makes a request to the third-party tracker Eulerian through *f7ds.liberation.fr*, the request is blocked.

We assess the efficiency of these blocklists as countermeasures. We use *Adblockparser* [101] that can parse Adblock Plus filters to directly match blocking list rules with all HTTP requests in the Alexa 1,739 sites that contain CNAME cloaking-based tracking in Table 3.3. Note that, *Adblockparser* has some limitations [102], but it does not impact our measurement for request-related to CNAME cloaking.

We inspect individual CNAME cloaking-based tracking URLs using these well-known blocklists in January 2020. The results of this experiment are shown in Table 3.6. We find that 2,713 CNAME cloaking-based tracking URLs have been flagged by these blocklists. This represents 77.87% of all CNAME cloaking-based tracking URLs in Alexa Top 300K sites. Besides that, the EasyPrivacy list detects almost as much CNAME cloaking-based tracking as combined lists. This means that CNAME cloaking domains detected by the Adguard tracking filter list are almost always detected by EasyPrivacy. Overall, tracker blocking lists thus do not effectively deal with CNAME cloaking-based tracking. Moreover, subdomains being used for CNAME cloaking may change often, which makes day-to-day blocklists updating tedious and time-consuming, and thus explain blocklist poor performances.

3.4.2 Browsers and extensions

Some browsers focus on security and privacy by blocking trackers. Browser extensions also use several techniques (such as blocklisting, or traffic monitoring) to block third-party tracking. We evaluate the ability of common browsers and extensions to block CNAME cloaking-based tracking.

We investigate five major browsers and six popular privacy-protecting extensions that support these browsers. We choose following popular browsers [61]: Chrome 80.0 [103], Opera 66.0 [104], Brave 1.4.92 [48], Firefox 73.0 [47] and Tor Browser 9.0.2 [49]. Regarding extensions, we use two criteria: blocking trackers and supporting multiple browsers. The privacy extensions that meet our criteria are Adblock 4.5.0 [105], Adblock Plus 3.7 [106], Privacy Badger 2020.1.13 [78], Disconnect 5.19.3 [45], Ghostery 8.4.6 [44], uBlock Origin 1.24.4 [46] and 1.24.5rc1 (developer's version) [107]. Ublock Origin 1.24.5rc1 has an anti CNAME cloaking-based tracking feature [107]. We include this version to provide an up-to-date picture of CNAME cloaking-based tracking countermeasures. We then collect all the HTTP requests and responses on the 1,739 websites containing CNAME cloaking-based tracking in Table 3.3. We use Atrica[3] [108], a multi-browser crawling library, to gather data on websites with CNAME cloaking-based tracking. To conduct a general comparison of browsers and privacy protection techniques, we crawl 1,739 websites using 40 different profile configurations (five browsers × eight extensions including the vanilla/bare setting). All the measurements were performed in March 2020 with three IP addresses in Japan. One crawling took approximately 4 to 6 hours on commodity hardware.

To reduce measurement error, we conducted three crawls and computed the relative standard error of the mean percentage of websites using CNAME cloaking-based tracking. We notice that there are also several possible sources of noise in our data. Some of these are internal and known, such as failure to connect to a website at a special time, or may also be external factors, such as network unreliability. To make a fair comparison, we set the website crawl timeout to 60 seconds. After this duration, if any website does not finish loading, we remove it and get the overlap among the three crawls of each profile.

Finally, we apply the same method (§ 3.2.2) to detect CNAME cloaking-based

[3] Atrica currently supports chromium-based and Firefox-based browsers.

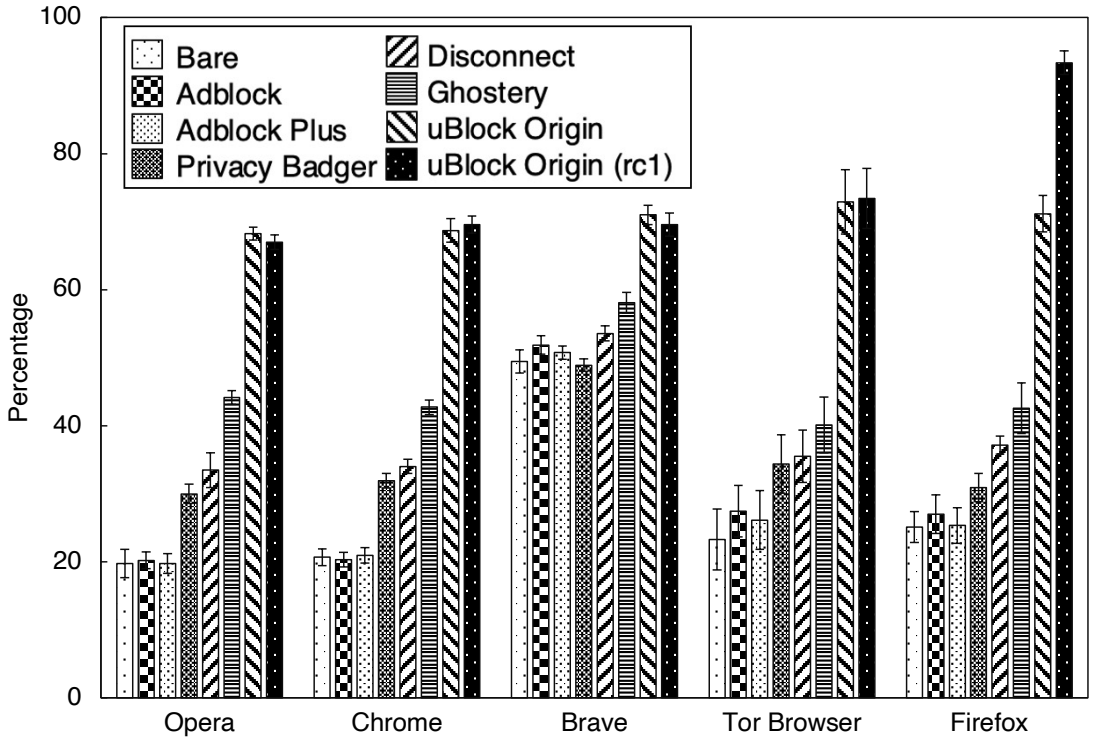


Figure 3.9: Detection performance of browsers and extensions regarding websites containing CNAME cloaking-based tracking. The mean and standard deviation are computed on three crawls.

tracking among these profiles.

Figure 3.9 shows the detection percentage of the CNAME cloaking-based tracking among browsers and their extensions. Overall, all browsers and extensions have a different impact on CNAME cloaking-based tracking. The most aggressive browser is Brave. It has the best performance among five browsers without any extension and blocks around 50% of websites that use CNAME cloaking-based tracking. We speculate that Shields feature is effective at detecting CNAME cloaking-based tracking. We also manually confirm that Shields blocks some CNAME cloaking-related subdomains, such as smetrics.10daily.com.au (Adobe), f7ds.liberation.fr (Eulerian), and 5ijo.01net.com (Eulerian).

For all browsers, the most effective extension is uBlock Origin that reduces around 70% of the websites containing CNAME cloaking. Adblock and Adblock Plus provide low protection abilities for all browsers. This result is not surprising because these

extensions target ad-blocking. Another notable point is that uBlock Origin version 1.24.5rc1 with anti-CNAME cloaking-based tracking technique is better than uBlock Origin version 1.24.4. It however only impacts to Firefox browser because other browsers do not provide an API that allows an extension to perform DNS lookups [51].

3.5 A machine learning approach for detecting CNAME cloaking-based tracking

Next, we describe our supervised machine learning-based approach to detect CNAME cloaking-based tracking.

3.5.1 Method overview

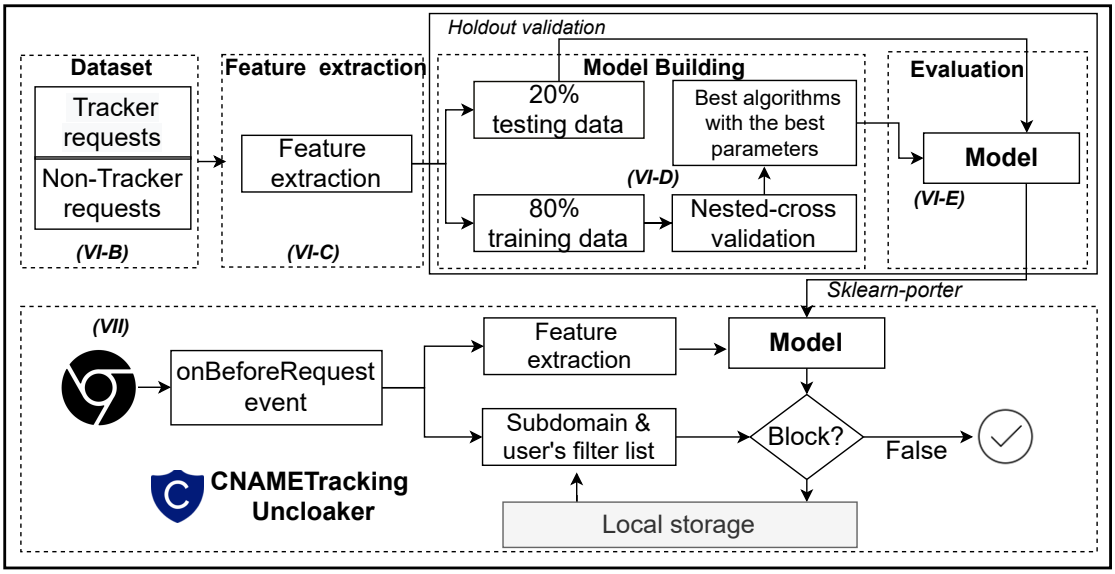


Figure 3.10: Overview of machine learning approach for detecting CNAME cloaking-based tracking requests and CNAMETracking Uncloaker browser extension management workflow.

Figure 3.10 shows an overview of our method consisting of four steps: data preparation, feature extraction, model development, and evaluation.

1. Select and divide the dataset into two sets, which we call the tracker requests and the non-tracker requests (§ 3.5.2).

2. Extract features for all requests by subdomain (§ 3.5.3).

3. Compare the F1 score of 10 classification algorithms using 10-fold stratified nested cross-validation with oversampled training data. After evaluating performance, we select the most effective classification algorithms with its best parameters to build a model (§ 3.5.4).

4. Evaluate the model with the testing data (§ 3.5.5.1).

3.5.2 Data preparation

We rely on 1,739 sites from Alexa Top 300K where CNAME cloaking-based tracking was previously detected § 3.3.3 and another 1,739 sites randomly picked from these 300K sites without CNAME cloaking-based tracking from. We label all requests as tracker instances and non-tracker instances.

To analyze the concept drift of our model (see § 4.7.3), we also pick up 43,429 subdomain-related requests which belong to 1,009 sites are related to CNAME cloaking-based tracking and 1,009 additional randomly picked sites without CNAME-cloaking from April 2018.

The details of the 2020 dataset and the 2018 dataset are listed in Table 3.7.

Table 3.7: Summary of data: 2,018 sites in 2018 and 3,478 sites in 2020.

Class	**April 2018**	**January 2020**
Tracker requests	2,490 (5.73%)	3,484 (10.01%)
Non-Tracker requests	40,939 (94.27%)	31,328 (89.99%)
Total subdomain-related requests	43,429 (100%)	34,812 (100%)
Total sites	2,018 (100%)	3,478 (100%)

3.5.3 Feature extraction

We experimentally extract the following features related to request linked to CNAME cloaking-related tracking.

- *method*: The desired action to be performed for a given request. We hypothesize that the GET method is usually used for subdomain-related requests linked to CNAME cloaking.
- *is_xhr*: The request uses an API that provides scripted client functionality for transferring data between a client and a server. We hypothesize that CNAME cloaking requires making HTTP requests in JavaScript between client and tracking provider server.
- *content_type*: The HTML tag that resulted in a request, such as image, javascript, or document, which are defined in this IDL [109]. We hypothesize that a specific resource is fetched in a web request for CNAME cloaking purpose (script).
- *len_url*, *len_sub*, and *len_prefix_sub*: The length of request URL, subdomain, and subdomain prefix. We hypothesize that there is a dissimilarity between the length of functional resources and CNAME cloaking resources.
- *num_prefix_sub*: The number of subdomain prefixes. We hypothesize that website's publishers use only one prefix to create a subdomain to deploy CNAME cloaking-based tracking.
- *prefix_sub_blacklist*: The subdomain prefix is among subdomain prefixes in tracking blocklists [79] [74]. We hypothesize that website's publishers use the same keyword (that is already in the blocklist) to create a subdomain to deploy CNAME cloaking-based tracking.
- *is_sub_dic*: The prefix of subdomain is a word in the English dictionary. We hypothesize that web publishers use random string as a subdomain to redirect to the tracking provider via CNAME record instead of meaningful keywords.
- *entropy_url*, *entropy_sub*, and *entropy_prefix_sub*: The randomness of request URL by calculating the metric entropy from request URL, subdomain, and

subdomain prefix. We hypothesize that there are differences in the metric entropy between functional request URL, subdomain, and subdomain prefix and CNAME cloaking resources.

3.5.4 Modeling and preliminary results

Using holdout validation method, we first split the 2020 dataset (Table 3.7) into testing data and training data. The percentage of the data held over for testing is 20%. It is used in § 4.7.3 to evaluate our model. Next, we describe how to build a classification model to detect CNAME cloaking-based tracking using testing data (80% of the 2020 dataset).

3.5.4.1 Model nested cross-validation

To perform hyperparameter optimization and model selection, while overcoming the problem of training dataset overfitting and the unbalanced nature of our dataset, we perform nested cross-validation using ADASYN algorithm [110]. We first use an outer 10-folds cross-validation loop to randomly split the training dataset into 10 smaller sets (folds) without replacement, where nine folds are used for the model training and the remaining one fold is for validating. We also use an inner loop to optimize the hyper-parameters of each model for each training dataset made of nine outer-folds. Note that, to evaluate the cross-validation with real data, we only conduct over-sampling on the minority class by applying ADASYN algorithm in the training folds and not in the validation folds. We perform a grid search optimization for this classification regarding the F1 score. After obtaining 10 performance estimates by repeating this procedure ten times, we take their average as the final performance estimate.

To deploy this machine learning model as a browser extension easily and effectively (see § 3.6), we first decide to compare 10 popular classification algorithms and evaluate their F1 score using above stratified nested cross-validation procedure on the training data.

We use the F1 score for evaluating the performance of the classifiers. Larger values of the F1 score (≈ 1.0) indicate better performance, and lower values (≈ 0) correspond to worse performance. Figure 3.11 shows the F1 scores for the 10 selected algorithms

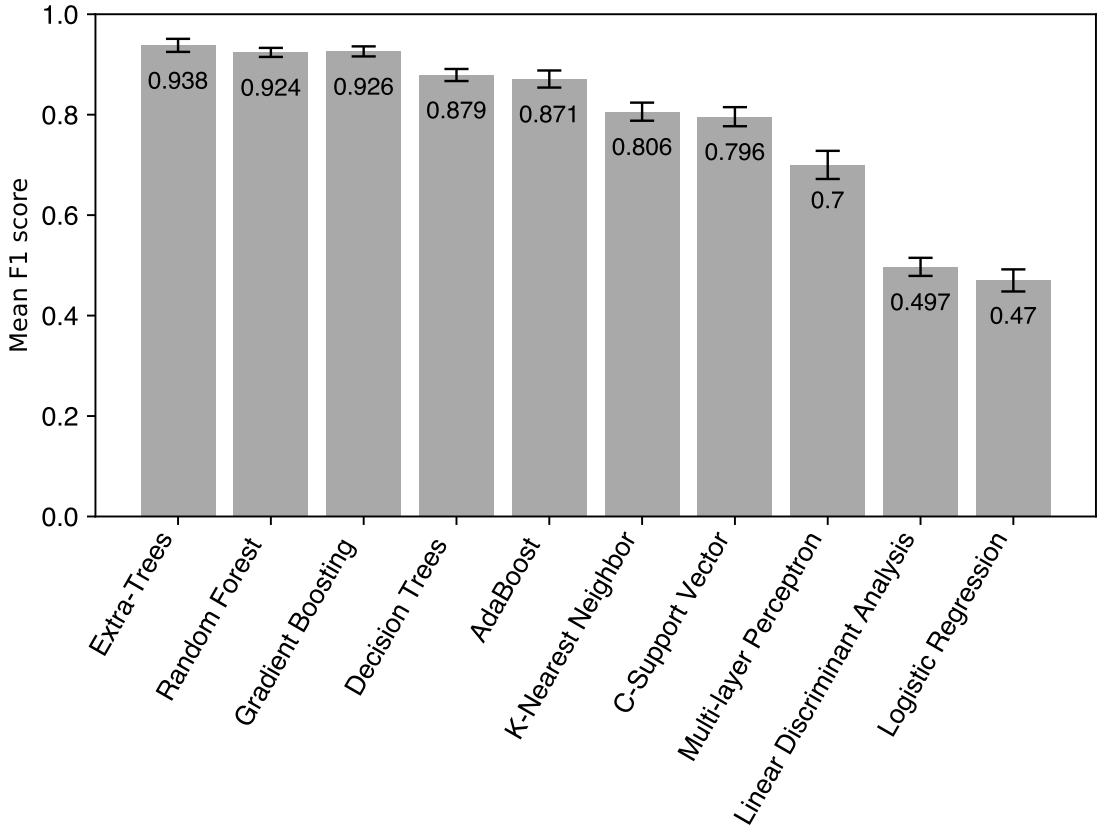


Figure 3.11: F1 score for the 10 selected classification algorithms using 10-fold stratified nested cross-validation in the dataset regarding the detection of requests linked to CNAME cloaking-based tracking. The mean and standard deviation are computed on the 10 folds of the nested cross-validation.

using 10-fold stratified cross-validation in the 2020 dataset for detecting requests linked to CNAME cloaking-related tracking. All classification algorithms have different detection performances. The most effective classification algorithm is Extra Trees, while Logistic Regression and Linear Discriminant Analysis classifiers show the worst performance for this dataset.

3.5.4.2 Selection of best algorithms and best parameters

From the previous performance evaluation, we select *Extra Trees* classifier and its set of best parameters (shown in Table 3.8) to train our model with oversampled training data.

Table 3.8: Best parameters of selected algorithm (Extra Trees) from the training phase regarding F1 score.

Algorithm	Parameter	Value
Extra Trees	max_features	10
	min_samples_split	2
	min_samples_leaf	1
	bootstrap	False
	n_estimators	100

3.5.5 Classification performance evaluation

3.5.5.1 The performance of model

The results obtained using the test set for requests linked to CNAME cloaking-based tracking detection on 20% of the 2020 dataset (preserved for this test) are shown in Table 3.9. We first show that our method detects requests related to CNAME cloaking-based tracking effectively. We achieve 0.941 of F1 score for tracker requests, 0.994 for non-tracker requests. We also obtain high precision and recall for both classes, which reduce the functional resources blocked by false positive, but still detect CNAME cloaking resources with less false negative.

By manually analyzing some false negatives and some false positives, we find that requests linked to CNAME cloaking have the same attributes as requests without CNAME cloaking-based tracking. For example, a script *https://ea.hofmann.es/eahof4645.js* that points to tracking provider *Eulerian*; its prefix *ea* not in the blocklists and it looks like a functional script. However, we can classify initiated the request by this script that actually perform tracking behavior as CNAME cloaking-based tracking. On the contrary, the request by subdomain *pf.newegg.com* is not used for CNAME cloaking. Its request URL contains detailed tracking of user actions (including browser, device, and IP location) that make the length of this URL request is similar to request-related to CNAME cloaking by tracking provider Adobe. However, this request is also blocked by the EasyPrivacy list.

Table 3.9: A comparison of detection performance.

Class	Method	Precision	Recall	F1 score
Non-Tracker	EasyPrivacy list	0.975	0.987	0.981
	Adguard filter list	0.938	0.996	0.966
	Blocklists+ DNS API (uBO)	1.000	0.984	0.992
	Our approach	**0.991**	**0.997**	**0.994**
Tracker	EasyPrivacy list	0.866	0.777	0.819
	Adguard filter list	0.926	0.410	0.568
	Blocklists + DNS API (uBO)	0.877	1.000	0.934
	Our approach	**0.970**	**0.914**	**0.941**

3.5.5.2 Comparison with other approaches

To block CNAME cloaking-based tracking without DNS resolution, well-known tracking blocklists such as the EasyPrivacy list and the AdGuard tracking filter list include the first-party subdomains which are fronts for CNAME cloaking. We thus compare the request detection performance between our machine learning approach and these well-known tracking blocklists in Table 3.9. We confirm lower F1 scores of tracker instances due to low recalls because of the lack of CNAME information for the tracking blocklists.

Furthermore, we simulate the performance of uBlock Origin with DNS API by applying CNAME resolution to the requests and then matching them with three tracking blocklists (EasyPrivacy, Peter Lowe's tracking, and uBlock Origin's own blocklists) as uBlock Origin does (blocklists + DNS API in the table). As expected, blocklists with DNS API achieve the best performance. The reason for this high performance is due to the absence of false negatives (i.e., recall is 1.0) which is not the case for other methods, though the number of false positives is small (i.e., precision is ≈ 0.9). The performance of our ML approach without DNS API is here close to the best performance thanks to the trained model.

3.5.5.3 Feature permutation importance

To discover discriminative features for the detection, we investigate the permutation importance [111] to calculate the feature importance of selected classifier for our

dataset. Note that, larger values indicate higher importance.

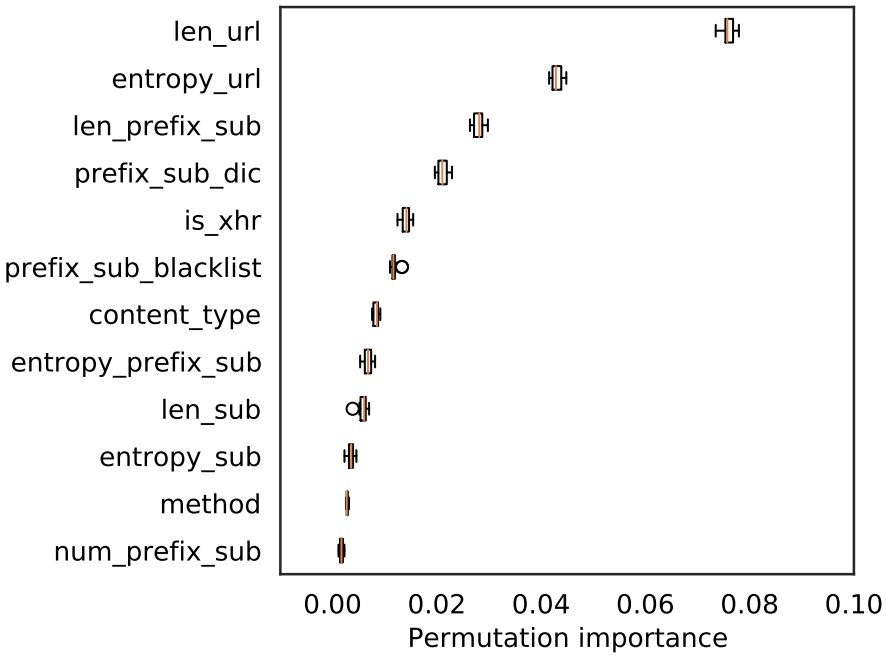


Figure 3.12: Permutation importance of the selected model for CNAME cloaking-related tracking occurring. The box extends from the lower to upper quartile values of the data, with a line at the median. The number of times a feature is randomly shuffled is n_repeats = 10.

Figure 3.12 shows the feature permutation importance of the model for detecting the requests. The result reveals that the request URL length (*len_url*) has the highest importance. We assume that almost all requests with a subdomain used for CNAME cloaking-based tracking have a length different than requests used for collecting site content. Besides, the randomness of URL request (*entropy_url*) and the subdomain prefix length (*len_prefix_sub*) are discriminative features for request detection. In addition to that, the subdomain prefix blocklist presence (*prefix_sub_blacklist*) is not effective to detect request-related to CNAME cloaking. This is due to the fact that some publishers also use the same subdomain prefix that is used for both CNAME cloaking and other non-tracking resources.

3.5.5.4 Concept drift analysis

Finally, we investigate the ability to detect requests related to CNAME cloaking-based tracking in the latest dataset (2020) using a model trained on the old dataset (2018). We use the 2018 dataset to train a model and test it on new sites collected in 2020 (Table 3.7). We apply the method explained in § 3.5.4 to build a model. Our result shows that the F1 score for CNAME cloaking-related requests detection is 0.703. Specifically, after two years, the F1 score decreases by 0.238. To explain this degradation, we examine the 2018 and 2020 datasets. In 2018, we do not see many random subdomain prefixes (*prefix_sub_dic*), while it is the case in 2020. These changes can be plausible reasons for the degradation of our model. Besides that, with rapid changes in web technology, tracking providers might also adjust their target site and change the implementation methods to deploy CNAME cloaking-based tracking Although the performance degradation is limited between 2018 and 2020, periodic model retraining can alleviate this problem if more detection accuracy is required.

3.6 CNAMETracking Uncloaker - a machine learning-based browser extension to protect end-user from CNAME cloaking-based tracking

Through the comprehensive analysis above, we demonstrate the effectiveness of our model to classify requests linked to CNAME cloaking-based tracking.

In this section, we propose *CNAMETracking Uncloaker*, a prototype of a Google Chrome browser extension that combines the blocklisting technique with the supervised machine learning for the automatic classification and filtering of CNAME cloaking-based tracking. As far as we know, only Firefox allows uBlock Origin to block CNAME cloaking by performing a DNS lookup of the hostname loading a resource. Other browsers do not support such DNS resolution API [46] (see § 2.3.1). Our prototype implementation thus circumvents the lack of DNS API in Chrome-based browsers.

3.6.1 Design and implementation

The objective of our method is to monitor the HTTP requests and block a request if it is related to CNAME cloaking-based tracking. To this end, we intercept all subdomain-related requests in-flight. In our extension, we track the HTTP requests and apply a machine learning model and a specific subdomain blocklist to detect and block CNAME cloaking-related requests. The novelty of our approach is a combination of the well-known technique of filtering with machine learning techniques to automatize the overall process.

Firstly, we use sklearn-porter [112] to transpile trained ExtraTree estimators (see § 3.5.4.2) to JavaScript. To remove all unnecessary characters from JavaScript source code without altering its functionality, we also minify this file using UglifyJS [113]. We also build a feature extraction module on the extension, which contains all 12 features described in § 3.5.3. Secondly, we build a CNAME cloaking-based tracking subdomain filter list based-on our dataset on § 3.5.2. When this blocklist is updated on the server-side, *CNAMETracking Uncloaker* obtains an updated blocklist for CNAME cloaking-related subdomains. Next, we implement the interface for accessing the user interface and changing the custom blocklist and allowlist. This interface allows end-users to construct customized lists according to the user's browsing habits; each user can have his/her different configuration. The end-users can also define an allowlist to add exceptions to reduce the effect of machine learning-related false positives. In the end, to intercept all HTTP requests by subdomain, we use the *onBeforeRequest* event, which is sent before any TCP connection is made and can be used to cancel or redirect requests. We then apply the feature extraction module and predict CNAME cloaking using the model. In addition, we match any subdomain-related URL request by CNAME cloaking-based tracking subdomain filter list and the custom filter lists of the end-user. If any subdomain-related request is not in the allowlist but it predicted as CNAME cloaking by model or by these blocklists, *CNAMETracking Uncloaker* blocks this request and keeps this element from being loaded onto the page by using a blocking event handler.

3.6.2 Performance evaluation

In order to measure the performance overhead of our extension, we compare it to the vanilla setting Chrome browser. We visit the top 50 websites according to Alexa's ranking (which do not contain CNAME cloaking-based tracking) and 50 websites linked to CNAME cloaking-based tracking in two cases: with and without *CNAMETracking Uncloaker*. During each visit, we record times of the *DOMContentLoaded* and *Load page* events. Each test is repeated 10 times with clean browsing history to retrieve the median duration. We use a commodity laptop computer with 8GB of RAM, Intel's i7 CPU, Ubuntu 18.04 and the latest version of Google Chrome (version 84.0). We present the performance overhead as the time difference between the median page load time with and without our extension.

Figure 3.13 (a) and (b) compare the overhead for *CNAMETracking Uncloaker* and the vanilla setting browser of the load events. We then plot two combinations: websites with blocked requests (blue circles) and without blocked requests (orange rectangles) by our extension. Our experiment shows that the overhead of *CNAMETracking Uncloaker* is acceptable. For the median gain for these sites, *CNAMETracking Uncloaker* extension introduces 0.08 seconds of median delay to the DOM loading. However, it is 0.408 seconds faster to the overall page loading, especially for websites that contain CNAME cloaking-based tracking. In particular, there are a limited number of websites without blocked requests (black circle) takes longer than the vanilla setting. Meanwhile, our extension blocks some resources (white rectangles) that reduce the time load site, this is why the overhead of *CNAMETracking Uncloaker* is lower than the other.

Interestingly, we observe that the vanilla setting is slower for three particular websites than *CNAMETracking Uncloaker*: *sohu.com* (12.15 seconds), *sina.com.cn* (7.54 seconds), and *clickavia.ru* (7.36 seconds). The reason for this behavior can be explained by the large number of HTTP requests that were blocked by our extension for these three sites: *sohu.com*, *sina.com.cn*, and *clickavia.ru* (seven, seven, and one request(s), respectively). Note that for *sohu.com*, these requests are not CNAME cloaking-based tracking. However, when we inspect these individual URLs using EasyPrivacy list as a reference, we can label these URLs as tracking-related. For *sina.com.cn*, one request is linked to tracking, but six requests are unrelated to CNAME cloaking. We also manually confirm that this website still works correctly, and that users can use our

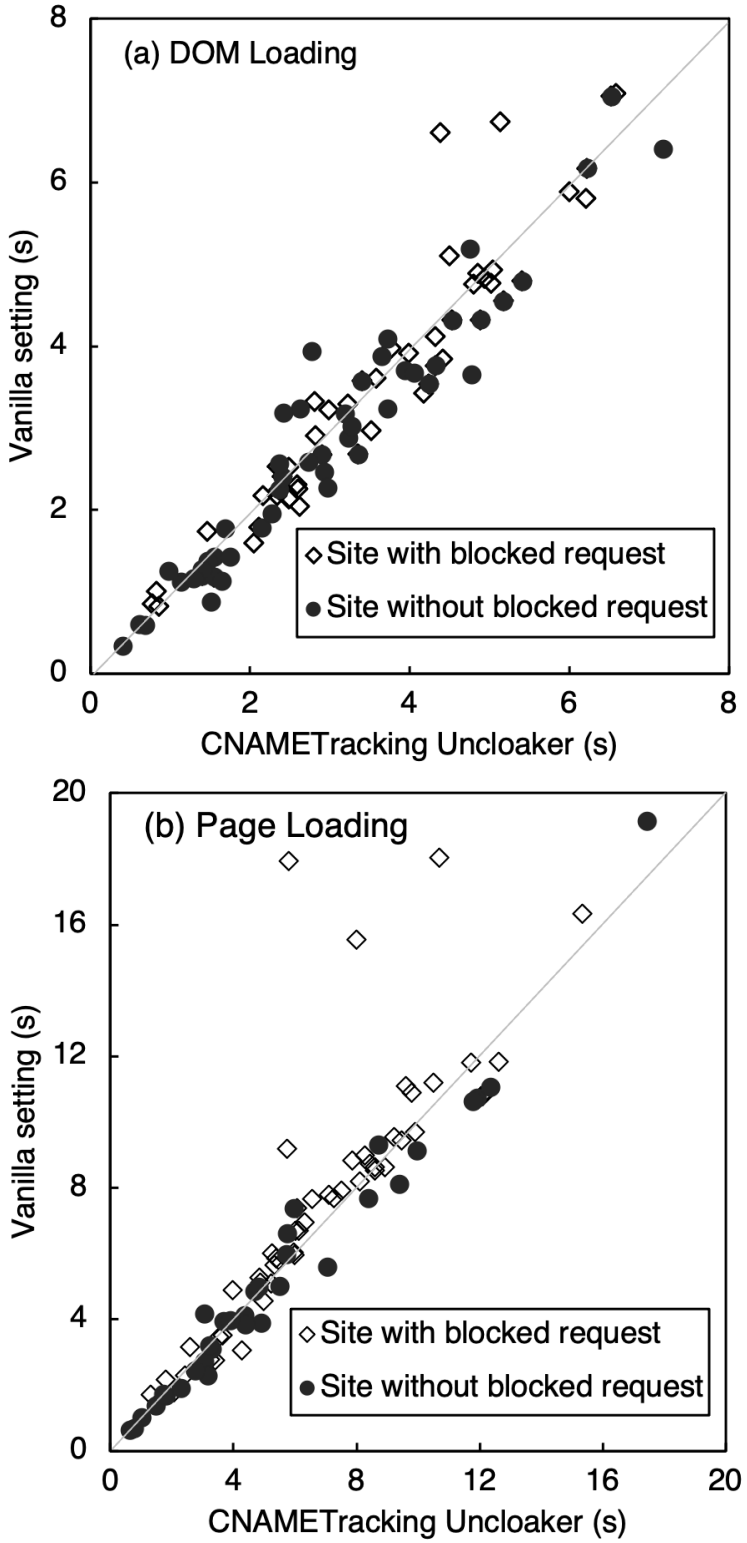


Figure 3.13: Comparison of the performance overhead by CNAMETracking Uncloaker and vanilla setting based on Alexa Top 50 websites and 50 websites containing CNAME cloaking-based tracking on median delay to the DOMContentLoaded event and page load overhead 10 times.

interface to customize the filter list (allowlist) to add an exception to our model.

In summary, *CNAMETracking Uncloaker* is able to filter out CNAME cloaking-based tracking from users' requests without significant performance degradation when compared with the default setting on Chrome browser. Note that *CNAMETracking Uncloaker* was only tested in our laboratory, we intend to do Beta testing of a group of target users to evaluate product performance in the real world in future work.

3.7 Summary

In this chapter, we characterized, detected, and protected the end-user against CNAME cloaking-based tracking on the web.

We conducted experiments to assess the occurrence and evolution of CNAME cloaking-based tracking. The results show that 1,739 websites in the Alexa Top 300K sites in January 2020 contain CNAME cloaking-based tracking. These websites are spread across many countries and categories. We also characterized a longitudinal analysis of CNAME cloaking-based tracking from 2016 to 2020. We found significant evidence that the top websites have injected more CNAME cloaking-based tracking in the last four years. The current best countermeasure to defend CNAME cloaking-based tracking, blocklist approach, is strongly depend on real-time name resolution. To overcome this limitation, we proposed a machine learning approach to detect HTTP requests containing CNAME cloaking-based tracking. Through the comprehensive analysis, we demonstrate the effectiveness of our method. Meanwhile, the DNS API being only available in Firefox browser, we developed a browser Chrome extension *CNAMETracking Uncloaker*, exploiting this machine learning-based approach to classify CNAME cloaking-based tracking. We performed an exhaustive evaluation of performance and effectiveness of our software prototype showing that machine learning-based techniques can be employed client-side as solutions into the browser to protect end-users against CNAME cloaking-based tracking.

Dataset availability: We provide a list of CNAMEs and tracking providers using CNAME cloaking-based tracking in our analysis at https://github.com/fukuda-lab/cname$_c$loaking. The extension is publicly available at Chrome Store: [114]. The raw crawled dataset will be also available from the authors on request.

4 PII leakage-based tracking

4.1 Introduction

PII leakage-based tracking refers to when a user's PII (i.e., email address, name, address) leaks to third-party domains (different from visited domains), to track user activities.

There are some existing methods to detect a part of PII leakage resources. Some common browsers focus on Referrer-Policy by removing the PII that can be leaked to third parties via referer header [71, 72]. Furthermore, Easylist [73], EasyPrivacy [74], and other filter lists update their blocklists, which can reduce the number of PII leakage to trackers and advertisers. However, this approach dramatically increases the size of the blocklists, and these filter lists need to be updated frequently which is tedious. Besides that, the Brave browser blocks tracking and advertising resources content from being loaded by default [75]. This browser however accounts for a small percentage of the browser market share [61, 76].

This work focuses on PII leakage, presents a persistent web tracking technique

based on this data transfer, and proposes a hybrid method to detect PII leakage by combining heuristic and supervised machine learning approaches. The main contributions of the paper are as follows. We first collect data on sign-up and sign-in flows completed like a typical user (§ 4.2). We then detect PII leakage to third-party domains (including CNAME cloaking) through four distinct methods even when that information is obfuscated (§ 4.3.1). Our results show that 42.3% of popular shopping-related first-party services leak PII to third-party receivers, the most common being Facebook (§ 4.3.2). We then conduct further experiments to investigate the impact of location to PII leakage and confirm that there are no significant differences compared to the usage of this phenomena in different country as *a new contribution*. Furthermore, we identify a tracking mechanism based on PII leakage that allows tracking providers to persistently identify users through cross-site, cross-browser, and cross-device tracking (§ 4.4.1). We then analyze the presence of PII leakage-based tracking and demonstrate that 20 tracking providers leverage PII so as to be able to tenaciously track user activities (§ 4.4.2). We also perform an extra experiment and confirm the Online Behavior Advertising occurring across multiple sites, browsers and devices based-on this tracking technique (§ 4.4.3). In addition, we look at the privacy policies of those 130 first parties to evaluate what disclosures are made to users about sharing PII to third parties (§ 4.5). Finally, we evaluate browsers and blocklists against leaked PII resources and confirm that there is still room for PII leakage countermeasures to improve (§ 4.6). Finally, we propose a hybrid method exploiting heuristic and machine learning for the detection of PII leakage (§ 4.7). Through the comprehensive analysis, we demonstrate the effectiveness of our method.

4.2 Data collection

4.2.1 Persona making

In order to build predictable PII-related strings, we first create a persona and its associated information that will be filled on websites' sign-up forms. This account persona contains the following information: username, name, phone, email address, date of birth, gender, job title, and postal address. We consider any information input by the user to be PII. We take inspiration from Refs. [65, 67, 115] to detect not only the

plaintext form but also the encoding/hashing form of leaked PII. We pre-compute a candidate set of tokens by applying all supported encodings, hashes, and checksums for each PII. Note that the encoding or hashing can be applied multiple times. Here we encode/hash each PII at most three times following previous work [67] to obtain a set of PII strings. We thus consider two different inputs that are used to build the PII-related strings that are detected in the next steps:

1. Any plaintext PII persona information.

2. A set of encodings, hashes, and their combination based on PII. The full list of supported hashes and encodings is given in Appendix § B.1.

4.2.2 Data acquisition

We first select websites that would be most appropriate for our work. We use the Tranco popular site list, which improves upon the shortcomings of the existing popular site lists: being unstable, having unreachable domain presence, and containing domains that are easily altered by an adversary [116]. From that, we obtain 404 shopping sites from the Tranco top 10,000 sites[1]. 95.0% of sites in this category contain authentication flows.

Next, to avoid the bias introduced by bots [117–120] and capture data precisely even when security mechanisms (such as bot detection and multi-factor authentication) are in place, we decide to collect our dataset by using a manual approach like that of a natural user as opposed to automated control. To form the basis for further investigation, we use the website's default settings to acquire the baseline data. In detail, we use a vanilla setting for Firefox 88 (Enhanced Tracking Protection was turned off) from an IP address located in Japan in May 2021. We always accept the default cookie settings for pop-ups (if any), but refuse all other types of solicitations, such as geolocation and notifications. We then browse the selected websites, fill all input fields in with the created persona information, press the submit button. After that, we open another browser and got the email confirmation link/code to activate the account successfully (if necessary), sign in with the created account, and reload

[1]We use the FortiGuard Web Filtering [95] dataset from May 2021 for the website category classification.

the site with a logged account. We also click a link (usually to a specific product) from the same domain to observe the event using leakage behavior on subpages compared with the homepage. During these processes, we collect HTTP requests (URLs, headers, and payload body - if any), HTTP responses (URLs and headers), and cookies (both those set/sent and a copy of stored browser cookies). Overall, after removing sites that were unreachable (22 sites), did not contain authentication flows (19 sites), and could not be signed up to due to the website's policy (56 sites[2]), we obtained successful authentication flows on 307 shopping sites, includes 68 sites that require email confirmation and 43 sites that use bot detection. These sites can not be crawled automatically even when effort has been made to match all complicated fields with the right information [68].

Regarding the ethical aspects of our research design and process, although we completed the authentication flows on target websites with a simulated persona, we believe that our procedure did not have any impact on their website operations.

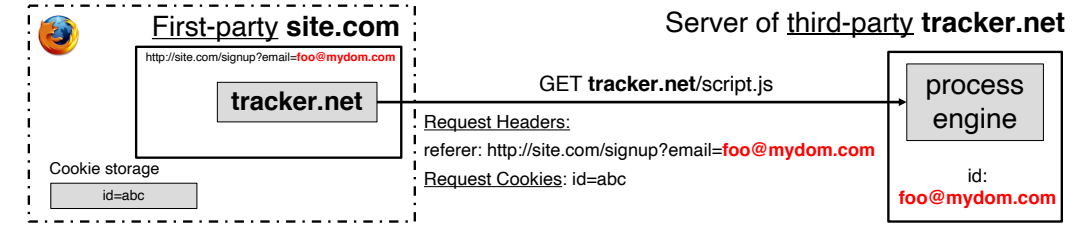
4.3 PII Leakage detection and analysis

4.3.1 PII leakage detection methods

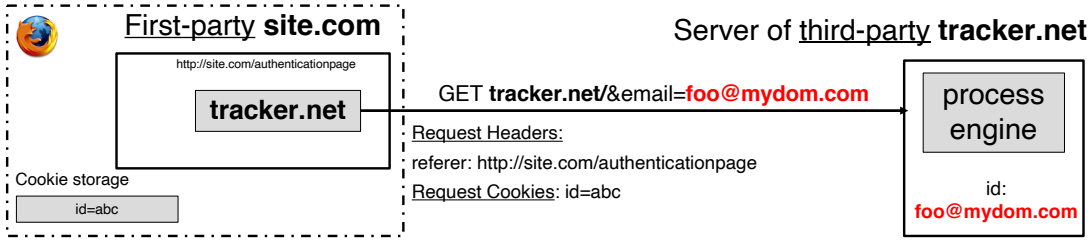
First of all, we separate the generic Top-Level Domain (gTLD) and country-code top-level domain (ccTLD) from the visited websites for all HTTP requests using the Public Suffix List [85] to separate first-party and *third-party* resources. In addition, to discover hidden third parties using *CNAME cloaking* (see chapter 3), we check CNAME records for each subdomain of the visited sites. We then match these CNAME answer sets with the CNAME cloaking blocklist by [121–123]. Finally, we combine these *CNAME cloaking* and *third-party* resources into the third parties and identify the following four methods that allow a first-party to send PII to a third-party.

Via referer header: A common method for PII leakage is based on the poorly coded sign-up and sign-in forms that expose sensitive information in URLs. More precisely, these forms submit user information to a first-party web server using the GET method. For instance, if *site.com* has a sign-up form that uses the GET method to submit a user's

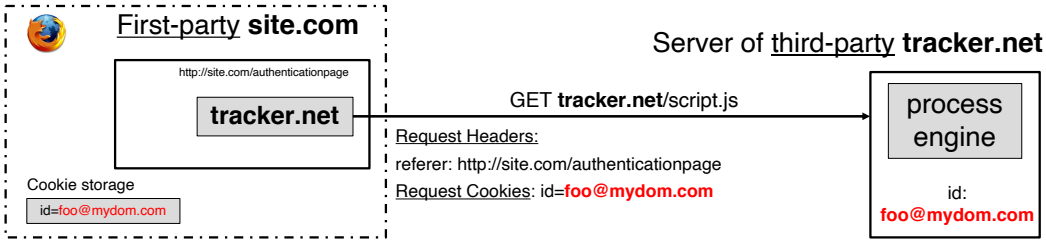
[2]Forty-seven sites required phone verification, six sites required special pieces of personal information such as identity documents, and three sites blocked the creation of new accounts for global customers.



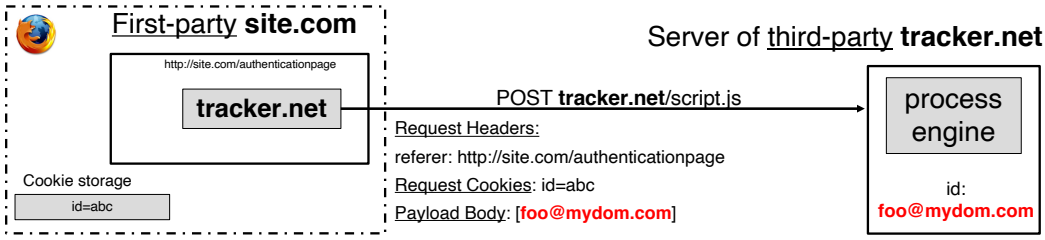
(a) Via referer header (as in [66–70])



(b) Via request URI (as in [66–70])



(c) Via cookie (as in [69])



(d) Via payload body (*newly detected*)

Figure 4.1: Four PII methods of leakage to third parties. This includes additional use of CNAME cloaking that was not considered before. PII is displayed in red.

email address, the user's browser creates a request with the form parameters containing the email address http://site.com/signup?&email=*foo@mydom.com*. In the case that *site.com* includes an external third-party, *tracker.net*, in its authentication flow, a request is sent to this third party. Because the *referer header* is sent with any request for a remote resource, this request will link to http://site.com/signup?&email=*foo@mydom.com* as the *referer header*. The PII provided by the user on the sign-up/sign-in form is thus sent to third-party domains (see Figure 4.1.a). We consider this method to be a form of unintentional leakage as in [66].

Via request URI: A request sent to a server can contain together with a set of parameters attached to the end of a URI, which structures additional information like PII for a given URI. For instance, the third-party *tracker.com* requests an email address and captures *foo@mydom.com* in the authentication flow. This URI is sent to *tracker.com* along with this information in the request URI (see Figure 4.1.b).

Via cookie: PII also can be leaked when third-party resources use PII to create cookie values. For instance, when *tracker.com* creates a cookie *id=foo@mydom.com*, this PII containing resource is sent to the third-party domain *tracker.com* (see Figure 4.1.c).

Via payload body: The last method involves PII being added to the payload body of an HTTP request and sent to a third-party domain. For instance, *tracker.com* will receive a request body from a first party with a payload that contains the parameter email address *foo@mydom.com* (see Figure 4.1.d).

Finally, we look for specific substrings as plaintext, encoded, hashed values and their combinations (§ 4.2.1), inside all third-party-related requests to detect any leaked PII. We are aware that the hashed string is a one-way algorithm that is impossible to convert into the original value. However, if the original email address appears in available email address lists, we can indirectly identify the individual email from the hashed string. So we still consider hashed email as PII throughout our study.

4.3.2 PII leakage analysis

We first focus on the PII leakage to third parties observed in the authentication flow of top shopping sites.

Overall, we detect the 130 first parties that leak PII among the 307 original authentication flows (through 1,522 requests that contain leaked PII). These first

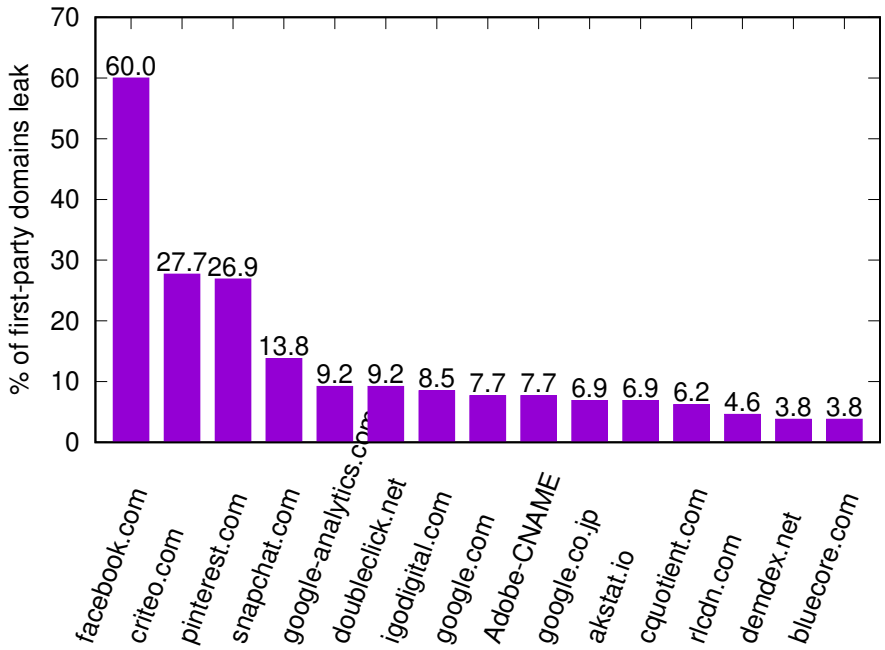


Figure 4.2: Top 15 third-party receiver domains involved in PII leakage included in 130 first-party senders from Tranco top shopping sites.

parties' PII leaks are sent to 100 third parties. Among these first parties, 46.15% send PII to at least three third-party domains, and the average number of third parties receiving PII was 2.97 per first-party site. We also observe 16 third-party receivers as the maximum from *loccitane.com*. We provide a breakdown of the top 15 third-party domains that received PII from the 130 first-party senders in the Tranco top shopping sites in Figure 4.2. We find that the domain *facebook.com* gets PII from 60.0% of the first-party senders. Regarding domain usage, while *Facebook*, *Criteo*, *Pinterest*, and *Snapchat* use a single domain to receive PII, *Google* and *Adobe* use multiple domains to receive PII from first-party senders.

4.3.2.1 PII leakage method

We break down the PII leakage to third parties according to the four methods in Table 4.1a. We observe seven accidental leakage cases in which three sites have a sign-up form that uses the GET method to submit a user's information data, and as a result, they leak PII to third parties *via referer header* (see Figure 4.1.a). Also,

Table 4.1: Breakdown of PII leakage to third parties. Percentages are given out of total of 130 first-party senders and 100 third-party leak receivers.

(a) By leaked method.

Method	#of Senders	# of Receivers
Referer header	3/2.3%	7/7.0%
URI	118/90.8%	78/78.0%
Payload body	43/33.1%	17/17.0%
Cookie	5/3.8%	1/1.0%
Combined	27/20.8%	8/8.0%

(b) By encoding/hashing

Encoding/hashing	#of Senders	# of Receivers
Plaintext	42/32.3%	56/56.0%
BASE64	19/14.6%	20/20.0%
MD5	35/26.9%	24/24.0%
SHA1	9/6.9%	6/6.0%
SHA256	91/70.0%	30/30.0%
SHA256 of MD5	2/1.5%	1/1.0%
Combined	21/16.2%	14/14.0%

(c) By PII type.

PII type	#of Senders	# of Receivers
Email	116/89.2%	94/94.0%
Username	1/0.8%	1/1.0%
Email,username	3/2.3%	6/6.0%
Email,name	29/22.3%	12/12.0%

there are five cases in which first-party senders use PII to create a cookie value that is automatically sent to the tracking provider Adobe via CNAME cloaking (see Figure 4.1.d). We discover that 126 first-party senders intentionally leak user's data to third-party receivers across multiple leaked methods. In addition, we find that 17 third-party domains receive PII via the payload body from 43 first-party senders that were not detected by observing the URL strings in previous work [67–70]. Furthermore, 27 first-party senders leak PII to 8 third parties by using combined methods (i.e., via request URI and cookie, via request URI and payload body). We hypothesize that the goal of this combination is to maximize the possibility of obtaining PII from first-party

sites in the context of third-party cookie restrictions and ad blocker utilization for privacy.

4.3.2.2 PII leakage by encoding/hashing

Next, we break down PII leakage by encoding/hashing to third parties in Table 4.1b. We first check whether any PII is sent as plaintext directly to the third-party domains. We find that 32.3% of the first-party senders directly send PII to 56% third-party receivers without any encoding or hashing function. In addition, 83.1% of first parties send PII as a hashed string to 49.0% of third-party domains. The most common one was SHA256 (70.0%). We also find two sites that use SHA56 of the MD5 hash string to send PII to the third-party domain *criteo.com*. In addition, we observed 21 first-party senders that leaked PII to third-party receivers by combining multiple encoding/hashing forms (i.e., plaintext and SHA256; BASE64, SHA1, and SHA256). PII leakage is much more complicated than previously reported [67–70].

4.3.2.3 PII leakage type

Finally, we analyze the nature of the leaked PII in Table 4.1c. The most common values that leaked were: email address (88.5%); username (0.8%); both email and username (1.5%); and email and name (22.3%). These PIIs are sufficient for correlating any user along with his or her browsing activity. Considering that third parties collect large amounts of data for a user, these types of information are valuable for the tracking ecosystem.

4.3.2.4 Email notification

When we conducted our experiments, the test email account started to receive email notifications from the visited sites. They mainly offered discounts or introduced new products. In total, we received 2,172 emails in the inbox and 141 emails in the spam folder. Notably, we have not yet received any emails belonging to any third-party domains involved in the PII leakage that were included in 130 first-party senders. We see that leaked PII is not used for third-party email marketing, but it could be used as an identifier to track user activity on the Internet.

4.3.3 Impact of geolocation to PII leakage

We also conduct extra experiments to evaluate the impact of geolocation on PII leakage occurring on websites.

To simulate users from different countries, we used a Virtual Private Network (VPN) provider with vantage points in different countries, including the United States and an EU country (Netherlands) where personal data is protected under the General Data Protection Regulation (GDPR). We then collect data (§ 4.2.2) on the 130 first-party sites that leak PII to third parties when we performed measurements from Japan. Finally, we apply the same method (§ 4.3.1) to detect PII leakage among these websites when crawled from different countries.

We find that eight first-party senders are impacted by the website location when sending PII to the third-party receiver *google.co.jp*. More specifically, this domain changes to *google.nl* in the Netherlands, and it does not appear in the US. We also observe that there is one site (*slickdeals.net*) on which we could not perform the sign-up flow in the Netherland because it does not support user account of EU/EEU citizens due to GDPR regulations. These results demonstrate that there is no significant effect of geolocation on PII leakage.

4.4 Persistent web tracking based on PII leakage

Here, we discuss the case where some third parties use PII to generate and store a unique persistent identifier of the user along with his/her browsing history for tracking purposes.

4.4.1 PII leakage-based tracking presumption

We first presume that PII can be a potential vector for persistent web tracking. A typical example of an HTTP request for persistent web tracking based on PII leakage is shown in Figure 4.3. The tracking provider *tracker.net* uses a PII identifier parameter (*trackid*) to capture information from first-party senders. After a user proceeds the sign-in flows on this site by submitting his/her information, a tracking script reads and pushes the user's email address *foo@mydom.com* as a value into the *trackid* parameter.

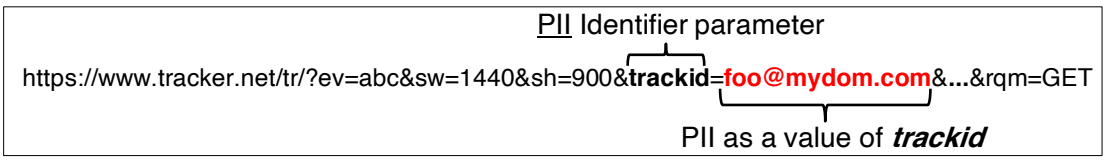


Figure 4.3: Example of HTTP traffic for persistent web tracking based on PII leakage. PII identifier parameter is different for each tracking provider, and PII value can be in hashing/encoding form.

This request is sent to the server of the tracking provider *tracker.net* along with this information. From there, the assumption is that *tracker.net* uses the *trackid* value *foo@mydom.com* as an identifier (*ID*) for user tracking. In case that a user uses this *ID* to complete the authentication flows in many sites, this method could be more effective than the traditional cookie-based approach because it can be used to identify user information on multiple sites, multiple browsers, and even multiple devices.

4.4.2 PII leakage-based web tracking cues

To confirm such a possibility, we first group the 130 first-party senders involved in the PII leakage together with the third-party receivers. Next, for each third-party receiver, we look for a PII identifier parameter (*trackid*) by checking the *parameter name* that assigns PII information as a *parameter value* in the URI parameters. We then list the PII identifier parameters per tracking provider. Then, to make sure that PII leakage is used for cross-site tracking, we focus on 34 third-party receivers involved in PII leakage that get the same ID from more than one first-party sender. Finally, to indicate the storage of a unique persistent identifier behavior, we consider 20 third-party receivers as tracking providers using PII leakage-based tracking when the ID appears not only in the authentication flows but also on the subpage of the first-party senders. This is an indisputable tracking behavior because these tracking providers appear on any subpages of the first-party senders along with the ID (PII), which can replace the usage of third-party cookies for tracking. Note that our experimental evaluation could have missed some tracking providers that appear only one time in the dataset (58 third-party receivers). We intend to expand our dataset in future work by using crowdsourced data collection to overcome this drawback.

Table 4.2: Breakdown of third-party receiver domains for persistent tracking based on PII leakage on popular shopping sites. All hashes are of full email address.

#	Receivers	# of Sender	Leaked Method	Encoding form	*trackid* Parameter
1	facebook.com	72	URI /Payload body	SHA256	udff[em]/ud[em]
		2	URI	MD5	ud[em]
2	criteo.com	26	URI	MD5	p0/p1
		4	URI	SHA256	p0
		5	URI	plaintext	p0/p1
		2	URI	SHA256ofMD5	p0/p1
3	pinterest.com	25	URI	SHA256	pd
		8	URI	MD5	pd
4	snapchat.com	18	URI/Payload body	SHA256	u_hem
		2	Payload body	MD5	u_hem
5	cquotient.com	7	URI	SHA256	emailId
6	bluecore.com	5	Payload body	BASE64	data
7	klaviyo.com	4	URI	BASE64	data
8	oracleinfinity.io	4	URI	SHA256	email_hash/ora*
9	rlcdn.com	4	URI	SHA1	s
10	adobe_cname	3	URI	SHA256	v*
11	castle.io	2	URI	plaintext	up
12	custora.com	2	URI/cookie	SHA1	uid/_custrack1_identified*
13	dotomi.com	2	URI	SHA256	dtm_email_hash
14	inside-graph.com	2	Payload body	plaintext	md
15	krxd.net	2	URI	SHA256	_kua_email_sha256
16	pxf.io	2	Payload body	SHA1	custemail
17	taboola.com	2	URI	SHA256	eflp
18	thebrighttag.com	2	URI	SHA256	_cb_bt_data
19	yahoo.com	2	URI	SHA256	he
20	zendesk.com	2	URI	BASE64	data

We show a breakdown of third-party receivers for persistent tracking based on PII leakage in Table 4.2. First, we notice that all tracking providers consistently use a PII type (email address) for tracking purposes. Obviously, the email address is an effective identifier that can reveal a user's identity. Second, we confirm that each tracking provider has at least one specific PII identifier parameter for multiple first-party senders. Third, the leaked PII (email address) is hashed in six encoding/hashing forms, mainly in SHA256. This raises a significant privacy concern for users when this ID can be used to share data among many tracking providers. Finally, we observed that some third parties received leaked PII in different encoding/hashing forms. In summary, PII can be used by third parties for user tracking not only on a single site but also for multiple sites, browsers, and devices when the user performs sign-up/sign-in in many

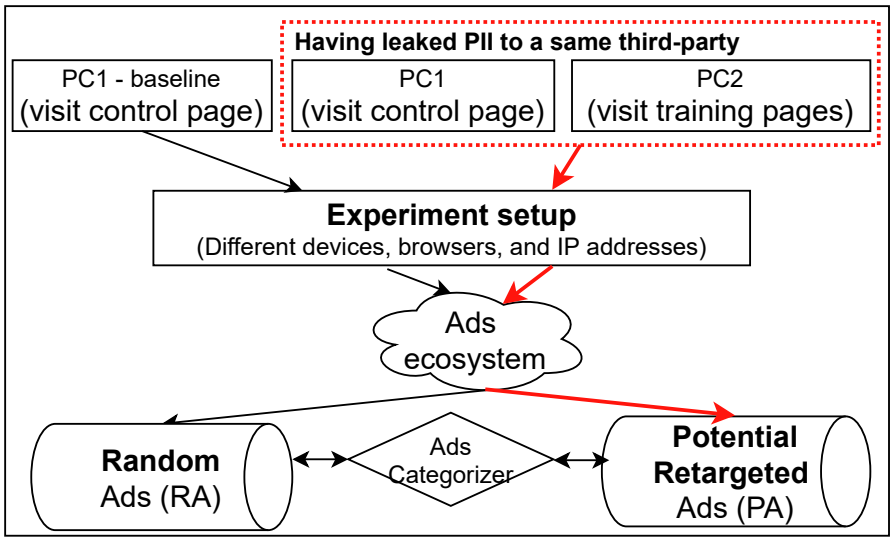


Figure 4.4: High-level description of the methodology to measure the effectiveness of cross-sites, cross-browser, and cross-device tracking using leaked PII by exploring Online Behavioral Advertising.

websites by the same PII.

4.4.3 PII leaked-based tracking confirmation

In this section, we perform experiments to evaluate the effectiveness of cross-sites, cross-browser, and cross-device tracking using leaked PII by exploring Online Behavioral Advertising (see Figure 4.4). Here, we design a methodology similar to [11–13] but modify it to take into account PII leakage occurring during manual signin.

4.4.3.1 Select training pages and control page

We first identify the input for our experiment.

Training Pages selection: We first select a set of shopping websites that leak PII to one of 20 third parties from Table 4.2 as training pages for a corresponding persona. In order to induce advertisers to associate a user with a specific interest, we must use enough distinct training pages. For this reason, we only consider the cases when the third-party receivers received PII from at least four first-party senders. By doing so, we keep nine groups of shopping websites where each group contain first-party senders

that leak PII to one of the following third-party receivers: *facebook.com*, *criteo.com*, *pinterest.com*, *snapchat.com*, *cquotient.com*, *bluecore.com*, *klaviyo.com*, *oracleinfinity.io*, and *rlcdn.com*.

Control Pages selection: We need a popular page that provides advertising space. Also, it must contain the authentication flows that leak PII to at least one of the nine third-party receivers above. We first select the popular news and media sites from Tranco Top 1000 sites and apply the same method (see § 4.3.1) to detect PII leakage to these third parties. We find that *newyorker.com* meets our requirement; it leaks PII to *rlcdn.com*, which belongs to Liveramp corporation.

4.4.3.2 Experiment Setup

Since we focus on the possibility of PII leakage-based tracking, we assume that the emulated user does not visit control pages and training pages in the same devices, browsers, and even IP addresses. Thus, there is no common identifier belonging to other tracking techniques shared between them [41, 124]. Here, we want to ensure that OBA is only triggered by PII leakage that occurs in the authentication flows of both the single control page and training pages.

We use two hosts with different IP addresses in Japan during our experiment. We first use the vanilla setting Firefox browser on a Ubuntu 20.10 desktop (PC1) to visit the control page *newyorker.com* 30 times and record all displayed advertisements without signing in. We call these ads as *random ads* or RA and keep them for a comparison later on. Secondly, we perform the authentication flows on the control page.

Next, using this email address, we complete authentication flows on the pool of training sites that leak PII to *rlcdn.com* in the vanilla setting Safari browser on a MacOS 12.0.1 desktop (PC2). We then visit the training pages (both the homepage and 7 articles on each site that are related to fashion) so that we allow the third-party domain *rlcdn.com* present in those pages to classify our persona with a very specific interest according to our deliberately narrow browsing behavior. From there, we visit a control page, *newyorker.com*, which allow us to collect the advertisements shown to our persona, which we call *potential retargeted ads* PA to later study whether they are driven by OBA. According to Liveramp documentation [125], we also set a waiting time to one day between the training phase and control phase for injecting a clear signal

over noise, from the training page to the ad ecosystem. Finally, we get the intersection, $RA \cap PA$, to filter out the same ads between the *random ads* and the *potential retargeted ads*; they are clearly not related to PII leakage-based tracking. We manually analyze the remaining ads in order to measure the ad topic similarity when ads are collected from distinct devices/browsers on websites that leak PII to the same third-party receivers. The created persona exhibits an interest to fashion shopping pages. We thus identify OBA occurrence as the presence of fashion-related ads that target the created persona.

4.4.3.3 Results

Overall, we collect 78 unique *random ads* RA and 82 *potential retargeted ads* PA on control pages *newyorker.com*. By comparing the ad landing page URL, which is the URL of the webpage accessed when one clicks on an ad, we first exclude 52.4% *potential retargeted ads* that are similar to *random ads* in our measurement. We then confirm that the 74.4% of remaining ads in the control page are related to fashion. Whereas, only 31.4% of random ads not inside the intersection $RA \cap PA$ are related to fashion. Thus, we observed the OBA due to PII leakage cross-site, cross-browser, and cross-device tracking. These advertisements belong to *doubleclick.net* and *criteo.net*. Obviously, the tracking providers and advertisers are integrated to maximize the benefit of user interests for target advertisements.

4.5 Transparency

In this section, we look at the privacy policies of the 130 first-party sites to evaluate their disclosure regarding PII sharing to third parties.

First, the 130 first-party sites that leak PII to third parties obtain user consent in the authentication flow as a mandatory requirement. However, they all give users a form to sign in order to indicate agreement without showing them the policy. As a result, *most end-users skip the policy altogether* [126]. This is a type of dark pattern that manipulates the user to agree to a privacy policy without user awareness. On the other hand, as shown in Table 4.3, we observe that all of the sites disclose that they collect personal information from users, and 111 sites state that they share or express degrees of the possibility of sharing personal information to third-party. However, *only nine*

Table 4.3: Privacy policy disclosures of 130 first-party senders that leak PII to third parties.

Disclosure		Number/percentage
Disclose PII sharing	Not specific	102/78.5%
	Specific	9/6.9%
No description of PII sharing		15/11.5%
Explicitly disclose PII NOT shared		4/3.1%
Total		**130/100%**

sites provide a list of third parties that receive this information. When provided, third party receiver lists are correct. Finally, some policies are very ambiguous; they do not mention PII sharing or even state that PII is used. In particular, 15 sites do not even mention PII sharing, and four sites declare that they do not share users' personal information with third parties for their marketing purposes.

4.6 In-browser protection techniques effective against PII leakage?

4.6.1 Evaluating browser countermeasures

Some browsers focus on security and privacy by blocking trackers. Brave provides a Shields feature that blocks third-party cookies, third-party storage, and blocks or replaces requests for tracking-related JavaScript. Safari has an Intelligent Tracking Prevention (ITP) feature that blocks third-party cookies, and partitions third-party storage by default. Also, Firefox restricts third-party cookies and storage for known trackers by default by introducing Enhanced Tracking Protection (ETP). We evaluate the ability of these browsers (vanilla/bare setting) to block PII leakage. We choose the following popular browsers [61]: Chrome 93 [103], Opera 79.0 [104], Safari 14.03, Firefox 92.0 [47], and Brave V1.29.81 [48]. Note that, we can miss some browsers that account for a small percentage of the browser market share [61, 76]. We then obtain data (see § 4.2.2) on the 130 first-party sites that leak PII to third parties. Finally, we apply the same method (see § 4.3.1) to detect PII leakage among these profiles.

Overall, we confirm that only the Brave browser has an impact on PII leakage detection. This browser accounts for a small percentage of the browser market share [61, 76]. The Safari browsers reduce three first-party senders and seven third-party receivers that receive PII via referer header method [72]. That is due to impact of the origin policy which cuts off referer header for all third-party requests in this browser. The remaining browsers do not solve this problem at all. For the Brave browser, the number of first parties leaking PII is reduced by 93.1%, and the number of third-party receivers is reduced by 92%. Brave misses eight third-party domains involved in PII leakage[3]. Also, we observe that there is one site on which we could not complete the sign-up flow by using Brave due to Shields blocking CAPTCHA verification (*nykaa.com*). These results show that this feature of Brave is still not able to completely block PII leakage and that there is thus room to improve protection.

4.6.2 Evaluating Blocklist-based countermeasures

Existing blocklists target advertisement and tracking resources and are utilized by browser extensions that intend to protect user privacy online. These lists may thus be able to block PII leakage resources provided by third parties. Here we assess their efficiency as countermeasures. We consider Easylist [73] and EasyPrivacy list [74]. EasyList is a blocklist that removes ads from web pages and it is used by popular browser plugins. EasyPrivacy is an optional supplementary blocklist that removes trackers from websites. We use *Adblockparser* [101] to evaluate the effectiveness of these two major blocklists by matching URLs against them. To determine if a request would have been blocked by an extension utilizing these lists, we directly match the block list rules quoted above with 1,522 HTTP requests that contained leaked PII and all requests in their request initiator chains. We inspect individual URLs containing leaked PII using these blocklists from June 2021.

The results of this experiment are shown in Table 4.4. Firstly, the performance of the combined blocklist varies widely depending on the PII leakage method. It is the most effective against the leakage via cookie. It blocks all senders and receivers of this data leak method. Secondly, the EasyList blocks only 8.0% of PII leakage, and it even

[3]List of eight third-party receivers missed by Brave version V1.29.81: *aliyun.com*, *cartsync.io*, *gravatar.com*, *herokuapp.com*, *intercom.io*, *lmcdn.ru okta-emea.com*, *zendesk.com*

Table 4.4: Detection performance of common browsers and well-known filters.

	Leaked method	Browser					Extension		
		Chrome	Opera	Firefox	Safari	Brave	EasyList	EasyPrivacy	Combined
Sender	Referer header	0/0%	0/0%	0/0%	3/100%	3/100%	0/0%	2/66.7%	2/66.7%
	URI	0/0%	0/0%	0/0%	0/0%	110/93.2%	1/0.8%	89/75.4%	97/82.2%
	Payload body	0/0%	0/0%	0/0%	0/0%	42/97.7%	0/0%	38/88.4%	38/88.4%
	Cookie	0/0%	0/0%	0/0%	0/0%	5/100%	0/0%	5/100.0%	5/100.0%
	Combined	0/0%	0/0%	0/0%	0/0%	26/96.3%	0/0%	24/88.9%	24/88.9%
	Total	**0/0%**	**0/0%**	**0/0%**	**1/0.8%**	**121/93.1%**	**1/0.8%**	**95/73.1%**	**102/78.5%**
Receivers	Referer header	0/0%	0/0%	0/0.0%	7/100.0%	7/100%	1/14.3%	6/85.7%	6/85.7%
	URI	0/0%	0/0%	0/0%	0/0%	72/92.3%	7/9.0%	51/65.4%	58/74.4%
	Payload body	0/0%	0/0%	0/0%	0/0%	16/94.1%	0/0%	12/70.6%	12/70.6%
	Cookie	0/0%	0/0%	0/0%	0/0%	1/100.0%	0/0%	1/100.0%	1/100.0%
	Combined	0/0%	0/0%	0/0%	0/0%	7/87.5%	0/0%	6/75.0%	6/75.0%
	Total	**0/0%**	**0/0%**	**0/0.0%**	**4/4.0%**	**92/92.0%**	**8/8%**	**65/65.0%**	**72/72.0%**

seems to almost not impact first-party senders. These are not surprising results because this list is used for ad-blocking protection. Finally, 78.5% (resp. 72.0%) of first-party (resp. third-party) senders (receivers) could be blocked when using both EasyList and EasyPrivacy. Considering PII leakage-based tracking (see Table 4.2), we observe that these blocklists block almost all resources belonging to these third-party domains. However, we also observe that there are still three missed third-party domains that received leaked PII: *custora.com*, *taboola.com*, and *zendesk.com*. In summary, even though the blocklist-based countermeasures can only partially deal with PII leakage, they can help reduce the number of leaked PII resources.

4.7 A hybrid approach for detecting PII leakage

In this section, we describe our hybrid method by combining heuristic and machine learning approaches to detect PII leakage. We first emphasize that we found six first-party senders that unintentionally send PII to third-party services via cookie by using CNAME cloaking, but many techniques have been developed to protect end-users from this tracking technique [62, 121, 122, 127], so we do not consider these six CNAME cloaking resource in our approach. Also, we discard 12 cases that leak PII via referer header and eight cases that leak PII via cookies because they can be easy to modify and block in a browser extension. Figure 4.5 shows an overview of our method consisting of two modules and an evaluation step to detect 1500 remaining requests-related PII leakage. Here are the next three steps and associated sections:

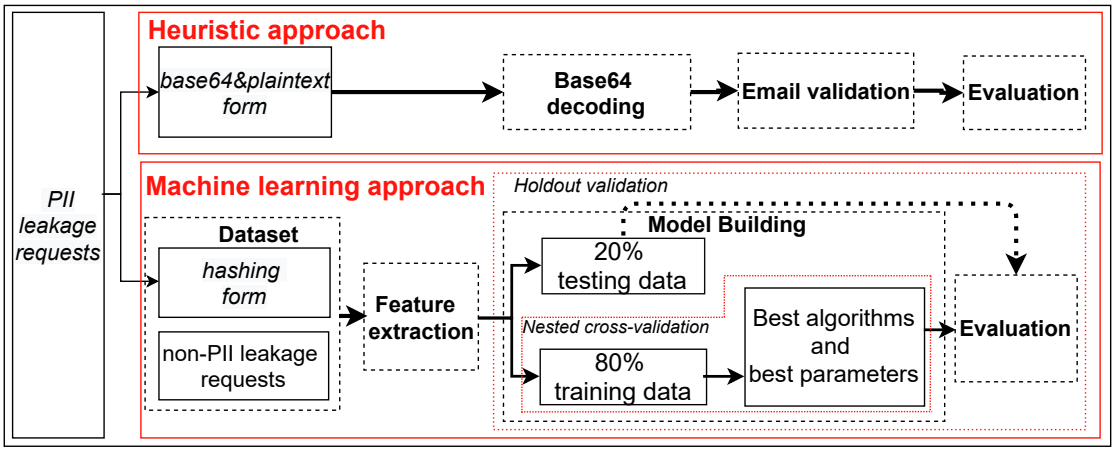


Figure 4.5: Overview of hybrid method for detecting PII leakage.

1. Heuristic approach building: Design a practical method to detect PII leakage in plaintext and base64 form (§ 4.7.1).

2. Machine learning approach building: Propose a supervised machine learning-based method to detect PII leakage in hashing form. (§ 4.7.2).

3. Evaluation: Evaluate and compare the performance of our method with other methods (§ 4.7.3).

4.7.1 A heuristic approach for detecting PII leakage in base64 and plaintext form

In this section, we present our heuristic approach for detecting PII leakage. Firstly, for each request related to PII leakage, we extract all sent parameters and their values, including parameters from the query string and payload body. We then decode base64 form from this piece of information.

Similar to the signup flow of web development, we apply email validation to verify if an email address is valid. Also, to make sure to not miss the URL-encoded format (or double URL-encoded format), we search the email provider domains from our dataset.

4.7.2 A machine learning approach for detecting PII leakage in hashing form

4.7.2.1 Data preparation

We rely on the third-party resources that belong to 304 shopping sites from Tranco Top 10K sites where PII leakage was previously collected § 4.2.2. The details of the dataset are listed in Table 4.5.

Table 4.5: Summary of dataset.

Class	Number	Percentage
PII leakage requests	1,086	1.85%
Non-PII leakage requests	80,481	98.15%
Total requests	81,567	100%

4.7.2.2 Feature extraction

We experimentally extract the features related to request linked to PII leakage. The features used for the classification are enumerated as below.

- *has_data*: The feature distinguishes the request URL containing the payload body.
- *method*: The desired action to be performed for a given request. We hypothesize that the GET and POST methods are usually used for requests linked to PII leakage.
- *unum*: The number of numbers appearing in an URL/payload data (if have). We hypothesize that requests linked to PII leakage contain a number of numbers by the hashing function.
- *qnum*: Number of the query field appearing in an URL/ payload data (if have). We hypothesize that requests linked to PII leakage contain a number of numbers by the hashing function.

- *anum*: Number of the key-value pairs appearing in an URL and payload data (if have). We hypothesize that requests linked to PII leakage contain a number of numbers by the hashing function.
- *url_len, data_len*: The length of request URL and payload data. We hypothesize that requests linked to PII leakage contain a number of numbers by the hashing function.
- *url_entropy, data_entropy*: The metric entropy of URL/payload data. We hypothesize that there is a significant difference in metric entropy among requests linked to PII leakage and others.
- *qhash*: True if a hash length appears in a key value of an URL/payload data, including 32 characters (MD5), 40 characters (SHA1), and 64 characters (SHA256). We hypothesize that requests linked to PII leakage contain a number of numbers by the hashing function.

4.7.2.3 Modeling and preliminary results

Using holdout validation method, we first split the dataset (Table 4.5) into testing data and training data. The training data is used to build the model, while the percentage of the data held over for testing is 20%, which is used to evaluate our model and is combined with the results from § 4.7.1 to evaluate this hybrid approach in the evaluation phase. Next, we describe how to build a classification model to detect PII leakage using training data (80% of the dataset).

Model nested cross-validation: To perform hyperparameter optimization and model selection, while overcoming the problem of training dataset overfitting and the unbalanced nature of our dataset, we perform nested cross-validation using ADASYN algorithm [110].

We first use an outer 10-folds cross-validation loop to randomly split the training dataset into 10 smaller sets (folds) without replacement, where nine folds are used for the model training and the remaining one fold is for validating. We also use an inner loop to optimize the hyper-parameters of each model for each training dataset made of nine outer-folds. Note that, to evaluate the cross-validation with the real data, we only conduct over-sampling on the minority class by applying ADASYN algorithm in the

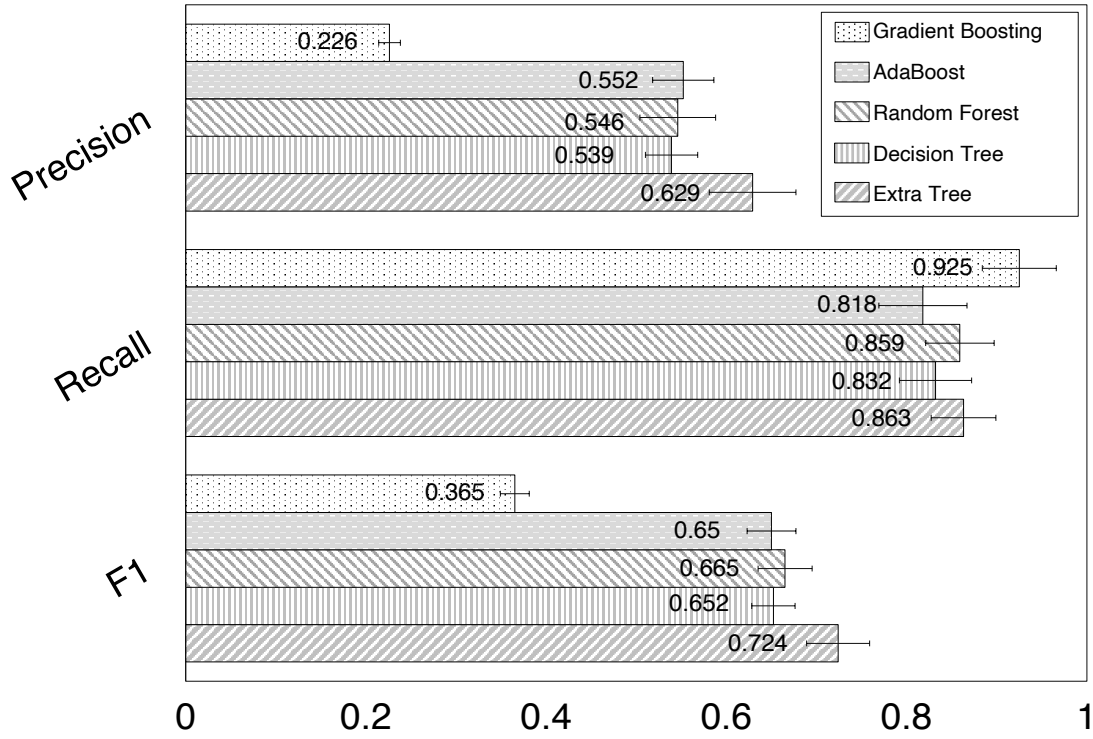


Figure 4.6: Precision, recall, and F1 score for the 5 selected classification algorithms using 10-fold stratified nested cross-validation in the dataset regarding the detection of request linked to CNAME cloaking-based tracking. The mean and standard deviation are computed on the 10 folds of the nested cross-validation.

training folds and not in the validation folds. We perform a grid search optimization for this classification regarding the F1 score. After obtaining 10 performance estimates by repeating this procedure ten times, we take their average as the final performance estimate. We then compare five popular classification algorithms: Gradient Boosting, Decision Tree, AdaBoost, Random forest and Extra Tree; and evaluate their precision, recall, and F1 score using the above stratified nested cross-validation procedure on the training data[4].

Selection of best algorithms and best parameters: We use the precision, recall, and F1 score for evaluating the performance of the classifiers. Larger values of the score (≈ 1.0) indicate better performance, and lower values (≈ 0) correspond to worse performance. Figure 4.6 shows the precision, recall, and F1 scores for the five selected algorithms

[4]A grid of parameter settings for each algorithm is given in Appendix Table B.1

Table 4.6: Best parameters of selected algorithm (Extra Trees) from the training phase regarding F1 score.

Algorithm	Parameter	Value
Extra Trees	criterion	gini
	max_features	10
	min_samples_split	2
	min_samples_leaf	1
	bootstrap	False

using nested 10-fold stratified cross-validation in the dataset for detecting PII leakage. In terms of precision and F1 score, the most effective classification algorithm is Extra Trees, while Gradient boosting classifier show the best overall recall score for this dataset. While a high recall means that a model's ability to find many PII leaks, a high precision ensures that detected leaks are indeed leaks to avoid disrupting website functionality (see § 4.6.1). Considering the fact that we want to minimize unnecessary website functionality loss, we decide to select the algorithm that obtains the best precision and acceptable recall to keep the website working properly. Therefore, we select the *Extra Trees* classifier and its set of best parameters (shown in Table 4.6) to train our model with oversampled training data.

4.7.3 Evaluation

4.7.3.1 The performance of the hybrid approach

The results obtained for requests linked to PII leakage detection are shown in Table 3.9, which combines the result of heuristic and machine learning approaches[5]. We first show that our hybrid method detects requests related to PII leakage effectively. We achieve 0.911 of F1 score for PII leakage requests detection. We also obtain high precision and recall, which reduce the functional resources blocked by false positive, but still detect PII leakage resources with less false negative.

[5]We detected all 414 requests link to PII leakage in the base64 and plaintext form. For hashing form, the performance of machine learning approach is given in Table B.2, Appendix § B.2.

Table 4.7: A comparison of detection performance.

Method	Precision	Recall	F1 score
EasyPrivacy list	0.051	0.752	0.095
Easy list	0.027	0.142	0.046
All (combined)	0.047	0.821	0.089
Brave	0.055	0.951	0.104
Our approach	**0.873**	**0.953**	**0.911**

4.7.3.2 Comparison with other approaches

To block PII leakage, existing well-known blocking lists, such as Easylist [73] and EasyPrivacy list [74] include the advertisement and tracking resources into their filter lists. We thus compare the request detection performance between our hybrid approach and these well-known tracking filter lists in Table 4.7. We confirm that lower F1 scores of tracker instances are due to low precision. That is because the tracking filter lists are not focusing on PII leakage but actually target advertisements and tracking resources. We thus argue that one should focus on the recall score to evaluate these techniques' ability to find PII leakage-related resources. As expected, our approach still achieves the best performance in terms of detecting PII leakage in general.

Furthermore, when comparing our hybrid method with the Brave browser, which achieves the best performance in browsers comparison (see § 4.6.1), we observe that the performance of our method is slightly better than Brave in terms of recall. Whereas intercept and monitoring the HTTP requests is not difficult, our countermeasures can easily integrate into browsers and extensions (see § 3.6).

4.8 Summary

In this chapter, we quantified the leakage of PII from the authentication flows in the Tranco top shopping sites and revealed a tracking technique that relies on this authentication. We showed that the PII leakage is more intricate than previously thought. In addition, we conducted experiments to assess the possibility of using leaked PII for persistent web tracking and evaluated this tracking technique in our dataset.

We pointed out the usage of leaked PII for cross-site, cross-browser, and cross-device tracking that has been deployed to personalize and target Online Behavioral Advertising. We also evaluated the ability of common browsers and well-known filter lists to detect PII leakage. Finally, we proposed a hybrid approach for detecting and thus protecting end-users privacy on the Internet.

Dataset availability: The lists of PII leakage URLs, first-party senders, and third-party receivers for popular shopping sites from the Tranco Top 10,000 sites (May 2021) are available at https://github.com/fukuda-lab/PII_leakage.

5 Discussion

This dissertation identified, detected, characterized two first-party cooperation-based third-party web tracking techniques, including CNAME cloaking-based tracking and PII leakage-based tracking. We observed that third-party web tracking is more sophisticated and pervasive than ever, and more advanced tracking techniques have been deployed. Further findings showed that the current protection techniques are ineffective against these tracking techniques. By applying heuristic and machine learning, our approach outperformed well-known tracking blocklists to protect end-user against CNAME cloaking-based tracking and PII leakage-based tracking.

5.1 Practical implications

This study provides several practical implications to researchers, browser vendors, website owners, and Internet users.

Firstly, we provided several contributions towards the ambitious goal of improving transparency of the web tracking ecosystem. After our original work, several studies

provided additional contribution regarding CNAME cloaking [128–130]. Aliyeva and Egele [128] and Ren et al. [129] focused on the security aspect by analyzing the impact of CNAME cloaking on browser cookie policies, which may transmit sensitive cookies to third parties. Furthermore, Dimova et al. [130] performed a large-scale longitudinal evaluation of CNAME cloaking-based tracking using the HAR dataset, and detected five tracking providers using a three-pronged approach and eight trackers from the CNAME cloaking blocklist by NextDNS [131]. Besides the security analysis, they presented results consistent with ours. They also found that the number of sites containing CNAME cloaking is gradually increasing over time. We thus argue that our initial contributions already have an impact on the research community. The research discoveries presented in this dissertation about CNAME cloaking have also led to a number of privacy improvements in web browsers and web standards, such as lowering the duration of cookies set in the HTTP response created through JavaScript to defend with CNAME cloaking by Safari [132].

Secondly, our approach for defending against third-party web tracking techniques can easily integrate into browsers and extensions, which make them become potential solutions to improve user privacy in the future.

Finally, we believe that this work helps to shed light on multiple tracking practices and bring more transparency to the web. Regarding end-users, our work will be helpful to raise awareness about online privacy and educate individuals on personal information protection [133]. For the online privacy research community, our work will open new research directions about privacy measurement and protection in the context of the generalization of countermeasures against third-party web tracking [134].

5.2 Recommendations

The privacy threats on the web are continuously evolving and presenting new challenges to privacy-enhancing tools. In this section, we provide recommendations to regulators, browser vendors, researchers, and end-users on how to improve transparency on the web.

Recommendation for regulator: As shown in our results, third-party web tracking is much more complicated and ambiguous than it is often regarded. Therefore, the

e-privacy regulations, which are a baseline to reinforce trust and security in the digital world, should be improved and extended. Regarding the processing of personal data, it will be reasonable when the data subject has given consent, which should be clearly and freely communicated, to the processing of his/her personal data for one or more specific purposes. Also, data processing must be done in a fair, and transparent manner by keeping personal data accurate, up-to-date, and secure. Whereas General Data Protection Regulation (GDPR) is the toughest privacy and security law in the world, it thus protects user privacy on the Internet for at least the European Union (EU) citizens. It would be helpful to increase user awareness and government engagement about processing personal data. Understanding about e-privacy of countries outside the EU, creating effective user surveys and building standard regulations to make a healthier internet will be necessary and effective work for regulators around the world.

Recommendations for browser vendors: Browser vendors are starting to take an active role in mitigating web tracking, and have been responsive to measurement results. We have seen sweeping privacy reform through built-in privacy protection features such as Firefox [47], Brave [48], and Tor Browser [49]. Chrome browser, which continues to maintain its market dominance, has notified that it will phase out third-party cookies in 2023. However, Google also introduces Manifest V3 [135] that will restrict the capabilities of web extensions, especially those that are designed to monitor, modify, and compute alongside the conversation between user browser and visiting websites. Obviously, with the substantial economic benefits from tracing user activities, there is a wide variety of viewpoints about user privacy among browser vendors. Given the current state of privacy concerns, we are hopeful that browser vendors will choose to stand on the side of protecting users. Besides, according to the detection performance of our machine learning approach against first-party cooperation-based third-party tracking, we believe that browser vendors can consider applying this approach to improve their in-browser privacy features in the future.

Recommendation for researchers: Research can reveal privacy risks and their exploitation of web tracking techniques on the Internet. Whereas web privacy measurement has played a key role in restricting online privacy incursions [18], the research efforts involved in technological and legal protection of user privacy require

more exploration. Considering that our measurement procedure can be generalized to detect and characterize third-party tracking techniques, researchers can apply our procedure to provide a deeper understanding about the web tracking mechanisms. We believe that we should work together to solve the privacy issues for a healthy internet.

Recommendation for user: There are massive tracking providers that collect user information on the Internet for various purposes. In this context, one of the most important initiatives is raising user awareness about the importance of their privacy. Although there is an ongoing debate about if individuals should be responsible for their data protection [136], the users should take an active role by educating themselves in protecting their online privacy. For instance, the users can use the anti-tracking browsers and extensions to protect their privacy as their benefits are widely known in security-sensitive sectors [137].

5.3 Limitations

Although we have expanded a great deal of effort in our study, there are still some drawbacks.

For CNAME cloaking-based tracking measurement, our *CNAME tracking filter list* in blocklist-based detection approach may be incomplete. We rely on the Easy Privacy list and Adguard Tracking filter list that are well-known and widely-used over the years, both by end-users and as ground-truth in academic works [67, 138, 139]. However, if the tracking providers use new domains that do not belong to these filter lists, we might miss some instances. Comparing with the lists of trackers that are often disguised using CNAME that publish by Adguard [127] and NextDNS [131], we observe that we dismissed four tracking providers: MO Internet Group, GENIEE, TraceDock, and Lead Forensics. However, these tracking providers did not appear in our dataset. Also, we observed some unstable crawling results even in our three crawls per site for in-browser protection techniques comparison. We omitted unfinished crawling results due to timeout, but still there is a possibility to miss CNAME cloaking resources because of the packet loss or the rapid change of web content. Finally,our dataset was crawled from a single country (Japan or United States), so there is a possibility of geographical differences. For PII leakage-based tracking measurement, our definition

of a third-party receiver might introduce some false positives because some websites host content from multiple domains. It however does not have a significant impact on our measurement. Also, aiming at avoiding the bias introduced by bots [117–120] and capturing data precisely even when security mechanisms are in place, we collected our dataset using a manual approach like a natural user as opposed to automated control. It is a good approach but hardly a scalable one. We intend to use crowdsourcing to overcome this limitation in the future.

In addition, our machine learning framework that applies for both CNAME cloaking detection and PII hashing leakage detection requires duplicating the initial training data processes but with a newer set of crawled data inputs. Although our countermeasures can easily integrate into browsers and extensions, this data collection process is time-consuming during deployment phase. At the same time, the proposed method complements existing techniques for first-party cooperation-based third-party web tracking detection and it can help researchers, browser extensions developers, and blacklist maintainers to build highly effective systems against these tracking techniques. Once tracking providers know the existence of the proposed detection methodology, they could attempt to evade it. In future work, we intend to improve our detection performance.

6 Conclusion

This chapter concludes the study by summarizing the key research findings in relation to the research aims and questions and discussing the value and contribution thereof. It will also propose opportunities for future research.

6.1 Thesis contributions

In this dissertation, we conducted experiments to assess the occurrence of first-party cooperation-based third-party web tracking, which have not yet been investigated. They include CNAME cloaking-based tracking and PII leakage-based tracking. We investigated their tracking mechanism and distinguished their characteristics which brings more transparency to the web. We also evaluated the ability of common browsers and well-known blocklists to detect these techniques. Finally, we developed the countermeasures for detecting these tracking mechanisms and thus protecting end-users privacy on the Internet.

6.2 Future work

In the following, we identify a number of possible directions for future work to improve transparency on the web.

1. Discovering unknown third-party web tracking techniques by automated detection of Online Behavioral Advertising (OBA): There are potentially many currently unknown techniques that are unable to detect on the HTTP client-side measurement. Also, this client-side approach is insufficient to detect server-side information flows between Tracking/Advertising networks. Apart from that, the main functionality of web tracking is to provide targeted advertisements, but this strategy misses the actual power of web tracking, especially, the different impact of stateful and stateless tracking mechanisms. OBA is able to reveal information flows between ad exchanges by leveraging re-targeted ads [11–13], which can tackle these problems. However, OBA measurement currently requires human actions, which is difficult and time-consuming to perform as a large-scale measurement. For this purpose, developing an open-source tool for OBA measurement will greatly improve privacy-related transparency to users.

2. Detecting tracking resources using machine learning: It is necessary to investigate techniques that can discern the intent of tracking, including stateful and stateless tracking, allowing countermeasures to be applied without causing website breakage. Using web measurement and machine learning to automatically detect and classify web tracking resources, is a potential solution that greatly improves the effectiveness of browser privacy tools to minimize the functional blocking on a website.

3. Building an auto-login framework for web privacy measurement: Authentications on a website are becoming more common. The privacy measurements could result unrealistic, with the crawler observing possibly very different content than what a user would get after logging into a website. However, to the best of our knowledge, there exists no practical method for signing up to a website automatically. To get an in-depth understanding of the privacy landscape, It would be interesting to develop an auto-login tool for evaluating and measuring web privacy at a large scale before and after the authentication process.

Publications

List of the publications related to this PhD dissertation.

Journal papers (peer-reviewed)

1. **Ha Dao**, Johan Mazel, and Kensuke Fukuda, "PII-based tracking ecosystem: characterization and countermeasure", IEEE/ACM Transactions on Networking (TNET), 2022, *(Major revision)*.

2. **Ha Dao**, Johan Mazel, and Kensuke Fukuda, "CNAME Cloaking-based Tracking on the Web: Characterization, Detection, and Protection", IEEE Transactions on Network and Service Management (TNSM), 2021.

International conference papers (peer-reviewed)

1. **Ha Dao** and Kensuke Fukuda, "Alternative to third-party cookies: Investigating persistent PII leakage-based web tracking", ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT), 2021.

2. **Ha Dao** and Kensuke Fukuda, "A machine learning approach for detecting CNAME cloacaking-based tracking on the Web", IEEE Global Communications Conference (GLOBECOM), 2020.

3. **Ha Dao**, Johan Mazel, and Kensuke Fukuda, "Characterizing CNAME Cloaking-Based Tracking on the Web", IEEE/IFIP Network Traffic Measurement and Analysis Conference (TMA), 2020.

Bibliography

[1] A short history of the web. [Online]. Available: https://home.cern/science/computing/birth-web/short-history-web

[2] S. Guha, B. Cheng, and P. Francis, "Challenges in measuring online advertising systems," in *Proceedings of the 10th ACM SIGCOMM IMC*, 2010, pp. 81–87.

[3] N. Bielova, "Web tracking technologies and protection mechanisms," in *Proceedings of ACM CCS*, 2017, pp. 2607–2609.

[4] H. Wang, M. K. Lee, and C. Wang, "Consumer privacy concerns about internet marketing," *Communications of the ACM*, vol. 41, no. 3, pp. 63–70, 1998.

[5] (2016) 2016 truste/ncsa consumer privacy infographic - gb edition. [Online]. Available: https://www.trustarc.com/resources/privacy-research/ncsa-consumer-privacy-index-gb/

[6] General data protection regulation - gdpr. [Online]. Available: https://gdpr-info.eu/

[7] California consumer privacy act (ccpa). [Online]. Available: https://oag.ca.gov/privacy/ccpa

[8] (2020) Full third-party cookie blocking and more. [Online]. Available: https://webkit.org/blog/10218/full-third-party-cookie-blocking-and-more/

[9] (2019) Today' s firefox blocks third-party tracking cookies and cryptomining by default. [Online]. Available: https://blog.mozilla.org/blog/2019/09/03/todays-firefox-blocks-third-party-tracking-cookies-and-cryptomining-by-default/

[10] (2021) An updated timeline for privacy sandbox milestones. [Online]. Available: https://blog.google/products/chrome/updated-timeline-privacy-sandbox-milestones//

[11] R. Balebako, P. Leon, R. Shay, B. Ur, Y. Wang, and L. Cranor, "Measuring the effectiveness of privacy tools for limiting behavioral advertising," in *W2SP'12-SP*, 2012.

[12] J. M. Carrascosa, J. Mikians, R. Cuevas, V. Erramilli, and N. Laoutaris, "I always feel like somebody's watching me: measuring online behavioural advertising," in *Proceedings of ACM CoNEXT*, 2015, pp. 1–13.

[13] K. Solomos, P. Ilia, S. Ioannidis, and N. Kourtellis, "Talon: an automated framework for cross-device tracking detection," in *Proceedings of RAID*, 2019, pp. 227–241.

[14] ederal trade commission. self-regulatory principles for online behavioral advertising: Tracking, targeting, and technology. [Online]. Available: https://www.ftc.gov/sites/default/files/documents/reports/federal-trade-commission-staff-report-self-regulatory-principles-online-behavioral-advertising/p085400behavadreport.pdf

[15] P. A. Pavlou, "State of the information privacy literature: Where are we now and where should we go?" *MIS quarterly*, pp. 977–988, 2011.

[16] C. H. Lee and D. A. Cranage, "Personalisation–privacy paradox: The effects of personalisation and privacy assurance on customer responses to travel web sites," *Tourism Management*, vol. 32, no. 5, pp. 987–994, 2011.

[17] F. Belanger, J. S. Hiller, and W. J. Smith, "Trustworthiness in electronic commerce: the role of privacy, security, and site attributes," *The journal of strategic Information Systems*, vol. 11, no. 3-4, pp. 245–270, 2002.

[18] S. Englehardt *et al.*, "Automated discovery of privacy violations on the web," 2018.

[19] B. Ur, P. G. Leon, L. F. Cranor, R. Shay, and Y. Wang, "Smart, useful, scary, creepy: perceptions of online behavioral advertising," in *proceedings of the eighth symposium on usable privacy and security*, 2012, pp. 1–15.

[20] A. M. McDonald and L. F. Cranor, "Americans' attitudes about internet behavioral advertising practices," in *Proceedings of the 9th annual ACM workshop on Privacy in the electronic society*, 2010, pp. 63–72.

[21] Http state management mechanism. [Online]. Available: https://tools.ietf.org/html/rfc6265

[22] Shared objects. [Online]. Available: https://help.adobe.com/en_US/as3/dev/WS5b3ccc516d4fbf351e63e3d118a9b90204-7d80.html

[23] Web storage api. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API

[24] Indexed database api 2.0. [Online]. Available: https://www.w3.org/TR/IndexedDB-2/

[25] A. Soltani, S. Canty, Q. Mayo, L. Thomas, and C. J. Hoofnagle, "Flash cookies and privacy," in *2010 AAAI Spring Symposium Series*, 2010.

[26] P. Papadopoulos, N. Kourtellis, and E. Markatos, "Cookie synchronization: Everything you always wanted to know but were afraid to ask," in *Proceedings of WWW*, 2019, pp. 1432–1442.

[27] A. Yamada, M. Hara, and Y. Miyake, "Web tracking site detection based on temporal link analysis," in *Proceedings of IEEE AINA Workshop*, 2010, pp. 626–631.

[28] H. Metwalley, S. Traverso, and M. Mellia, "Unsupervised detection of web trackers," in *Proceedings of IEEE GLOBECOM*, 2015, pp. 1–6.

[29] X. Pan, Y. Cao, and Y. Chen, "I do not know what you visited last summer protecting users from third-party web tracking with tracking free browser," in *Proceedings of NDSS'15*, 2015, pp. 1–16.

[30] Q. Wu, Q. Liu, Y. Zhang, P. Liu, and G. Wen, "A machine learning approach for detecting third-party trackers on the web," in *Proceedings of ESORICS*, 2016, pp. 238–258.

[31] M. Ikram, H. J. Asghar, M. A. Kaafar, A. Mahanti, and B. Krishnamurthy, "Towards seamless tracking-free web: Improved detection of trackers via one-class learning," *Proceedings on PET*, vol. 2017, no. 1, pp. 79–99, 2017.

[32] B. Krishnamurthy and C. Wills, "Privacy diffusion on the web: a longitudinal perspective," in *Proceedings of WWW*, 2009, pp. 541–550.

[33] J. R. Mayer and J. C. Mitchell, "Third-party web tracking: Policy and technology," in *IEEE S&P*, 2012, pp. 413–427.

[34] F. Roesner, T. Kohno, and D. Wetherall, "Detecting and defending against third-party tracking on the web," in *Proceedings of USENIX NSDI*, 2012, pp. 12–12.

[35] T. Libert, "Exposing the hidden web: An analysis of third-party http requests on one million websites," *International Journal of Communication*, 2015.

[36] S. Schelter and J. Kunegis, "Tracking the trackers: A large-scale analysis of embedded web trackers," in *Proceedigns of AAAI ICWSM*, 2016.

[37] T. Urban, D. Tatang, M. Degeling, T. Holz, and N. Pohlmann, "Measuring the impact of the gdpr on data sharing in ad networks," in *Proceedings of ACM CCS*, 2020, pp. 222–235.

[38] P. Eckersley, "How unique is your web browser?" in *Proceedings of PETS*. Springer, 2010, pp. 1–18.

[39] F. Besson, N. Bielova, and T. Jensen, "Hybrid information flow monitoring against web tracking," in *Proceedings of the IEEE CSF 2013*. IEEE, 2013, pp. 240–254.

[40] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless monster: Exploring the ecosystem of web-based device fingerprinting," in *IEEE S&P*. IEEE, 2013, pp. 541–555.

[41] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel, "Fpdetective: dusting the web for fingerprinters," in *Proceedings of ACM CCS*, 2013, pp. 1129–1140.

[42] N. Nikiforakis, W. Joosen, and B. Livshits, "Privaricator: Deceiving fingerprinters with little white lies," in *Proceedings of WWW*, 2015, pp. 820–830.

[43] G. Merzdovnik, M. Huber, D. Buhov, N. Nikiforakis, S. Neuner, M. Schmiedecker, and E. Weippl, "Block me if you can: A large-scale study of tracker-blocking tools," in *IEEE EuroS&P 2017*, 2017, pp. 319–333.

[44] Ghostery makes the web cleaner, faster and safer! [Online]. Available: https://www.ghostery.com/

[45] Disconnect. [Online]. Available: https://disconnect.me/

[46] R. Hill. ublock origin - an efficient blocker for chromium and firefox. fast and lean. [Online]. Available: https://github.com/gorhill/uBlock

[47] (2004) Firefox browser. [Online]. Available: https://www.mozilla.org/en-US/exp/firefox/

[48] (2019) Brave browser. [Online]. Available: https://brave.com/

[49] Tor browser. [Online]. Available: https://www.torproject.org/

[50] (2008) The design and implementation of the tor browser [draft]. [Online]. Available: https://2019.www.torproject.org/projects/torbrowser/design/

[51] (2019) Address 1st-party tracker blocking #780. [Online]. Available: https://github.com/uBlockOrigin/uBlock-issues/issues/780#issuecomment-566845764

[52] O. Poitrey. (2019) Nextdns first to support blocking of all third-party trackers disguised as first-party. [Online]. Available: https://medium.com/nextdns/nextdns-added-cname-uncloaking-support-becomes-the-first-cross-platform-solution-to-the-problem-e3f437f84342

[53] R. Cointepas. (2019) Cname cloaking, the dangerous disguise of third-party trackers. [Online]. Available: https://medium.com/nextdns/cname-cloaking-the-dangerous-disguise-of-third-party-trackers-195205dc522a

[54] J. Leyden. (2019) Web trackers using cname cloaking to bypass browsers' ad blockers. [Online]. Available: https://portswigger.net/daily-swig/web-trackers-using-cname-cloaking-to-bypass-browsers-ad-blockers

[55] Adguard dns. [Online]. Available: https://adguard.com/en/adguard-dns/overview.html

[56] Pi-hole – a black hole for internet advertisements. [Online]. Available: https://pi-hole.net/

[57] K. Borgolte and N. Feamster, "Understanding the performance costs and benefits of privacy-focused browser extensions," in *Proceedings of WWW*, 2020, pp. 2275–2286.

[58] Adguard adblocker. [Online]. Available: https://adguard.com/en/adguard-browser-extension/overview.html

[59] Cname cloaking and bounce tracking defense. [Online]. Available: https://webkit.org/blog/11338/cname-cloaking-and-bounce-tracking-defense/

[60] What's brave done for my privacy lately? episode #6: Fighting cname trickery. [Online]. Available: https://brave.com/privacy-updates-6/

[61] (1998) Browser statistics. [Online]. Available: https://www.w3schools.com/browsers/

[62] ublock origin - firefox. [Online]. Available: https://addons.mozilla.org/en-US/firefox/addon/ublock-origin/

[63] (2003) Safari - apple. [Online]. Available: https://www.apple.com/safari/

[64] J. Ren, A. Rao, M. Lindorfer, A. Legout, and D. Choffnes, "Recon: Revealing and controlling pii leaks in mobile network traffic," in *Proceedings of ACM MobiSys*, 2016, pp. 361–374.

[65] O. Starov and N. Nikiforakis, "Extended tracking powers: Measuring the privacy diffusion enabled by browser extensions," in *Proceedings of WWW*, 2017, pp. 1481–1490.

[66] O. Starov, P. Gill, and N. Nikiforakis, "Are you sure you want to contact us? quantifying the leakage of pii via website contact forms," *Proceedings on PET*, vol. 2016, no. 1, pp. 20–33, 2016.

[67] S. Englehardt, J. Han, and A. Narayanan, "I never signed up for this! privacy implications of email tracking," *Proceedings on PET*, vol. 2018, no. 1, pp. 109–126, 2018.

[68] M. Chatzimpyrros, K. Solomos, and S. Ioannidis, "You shall not register! detecting privacy leaks across registration forms," in *Computer Security*. Springer, 2019, pp. 91–104.

[69] B. Krishnamurthy, K. Naryshkin, and C. Wills, "Privacy leakage vs. protection measures: the growing disconnect," in *Proceedings of WWW*, vol. 2, no. 2011, 2011, pp. 1–10.

[70] J. Mayer. (2011) Tracking the trackers: Where everybody knows your username. [Online]. Available: http://cyberlaw.stanford.edu/node/6740

[71] Firefox 87 trims http referrers by default to protect user privacy. [Online]. Available: https://blog.mozilla.org/security/2021/03/22/firefox-87-trims-http-referrers-by-default-to-protect-user-privacy/

[72] Preventing tracking prevention tracking. [Online]. Available: https://webkit.org/blog/9661/preventing-tracking-prevention-tracking/

[73] (2005) Easy list. [Online]. Available: https://easylist.to/easylist/easylist.txt

[74] (2006) Easyprivacy. [Online]. Available: https://easylist.to/easylist/easyprivacy.txt

[75] Grab bag: Query stripping, referrer policy, and reporting api. [Online]. Available: https://brave.com/privacy-updates/5-grab-bag/

[76] (2020) What is brave browser' s market share. [Online]. Available: https://www.ctrl.blog/entry/brave-market-share.html

[77] (2019) Data collection cnames and cross-domain tracking. [Online]. Available: https://docs.adobe.com/content/help/en/id-service/using/reference/analytics-reference/cname.html

[78] Privacy badger | electronic frontier foundation. [Online]. Available: https://www.eff.org/privacybadger

[79] Adguard tracking protection filter. [Online]. Available: https://filters.adtidy.org/extension/chromium/filters/3.txt

[80] The top 500 sites on the web. [Online]. Available: https://www.alexa.com/topsites

[81] J. Mazel, R. Garnier, and K. Fukuda, "A comparison of web privacy protection techniques," *Computer Communications*, vol. 144, pp. 162–174, 2019.

[82] S. Englehardt and A. Narayanan, "Online tracking: A 1-million-site measurement and analysis," in *Proceedings of ACM CCS*, 2016, pp. 1388–1401.

[83] Q. Scheitle, O. Hohlfeld, J. Gamba, J. Jelten, T. Zimmermann, S. D. Strowes, and N. Vallina-Rodriguez, "A long way to the top: significance, structure, and stability of internet top lists," in *Proceedings of ACM IMC*, 2018, pp. 478–493.

[84] Http archive. [Online]. Available: https://httparchive.org/

[85] (2007) Public suffix list. [Online]. Available: https://publicsuffix.org/list/

[86] Rapid7 open data, forward dns (fdns). [Online]. Available: https://opendata.rapid7.com/sonar.fdns$_v$2/

[87] Dnsdb database. [Online]. Available: https://www.dnsdb.info/

[88] Marketing attribution and data management - eulerian. [Online]. Available: http://www.eulerian.com/

[89] S. Kayan. cdnfinder. [Online]. Available: https://github.com/turbobytes/cdnfinder/blob/master/assets/cnamechain.json

[90] W. Ma. china-cdn-domain-whitelist. [Online]. Available: https://github.com/mawenjian/china-cdn-domain-whitelist/blob/master/china-cdn-domain-whitelist.txt/

[91] Z. Weinberg, S. Cho, N. Christin, V. Sekar, and P. Gill, "How to catch when proxies lie: Verifying the physical locations of network proxies with active geolocation," in *Proceedings of ACM IMC*, 2018, pp. 203–217.

[92] Ip geolocation api. [Online]. Available: https://ip-api.com/

[93] Free ip geolocation api. [Online]. Available: https://freegeoip.app/

[94] Maxmind geoip2 python api. [Online]. Available: https://dev.maxmind.com/geoip/geoip2/geolite2/

[95] (2005) Fortiguard web filtering. [Online]. Available: https://fortiguard.com/webfilter

[96] Tracking protection lists - trackers we block. [Online]. Available: https://github.com/mozilla-services/shavar-prod-lists/blob/master/disconnect-blacklist.json

[97] Pardot —b2b marketing automation. [Online]. Available: https://www.pardot.com/

[98] Act-on — exceptional marketing automation software. [Online]. Available: http://www.act-on.com/

[99] Oracle eloqua marketing automation. [Online]. Available: https://www.oracle.com/cx/marketing/automation/

[100] Webtrekk. [Online]. Available: https://www.webtrekk.com/

[101] (2016) Python parser for adblock plus filters. [Online]. Available: https://github.com/scrapinghub/adblockparser

[102] Adblockparser limitations. [Online]. Available: https://github.com/scrapinghub/adblockparser#limitations

[103] (2008) Chrome browser. [Online]. Available: https://www.google.com/chrome/

[104] (1995) Opera browser. [Online]. Available: https://www.opera.com/

[105] Adblock. [Online]. Available: https://getadblock.com/

[106] (2010) Adblock plus. [Online]. Available: https://adblockplus.org/

[107] R. Hill. ublock origin - developer build 1.24.5rc0. [Online]. Available: https://github.com/gorhill/uBlock/releases/tag/1.24.5rc0

[108] V. Gerest. Atrica. [Online]. Available: https://github.com/fukuda-lab/atrica

[109] base - mozsearch. [Online]. Available: https://searchfox.org/mozilla-central/source/dom/base/

[110] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proceedings of IEEE IJCNN*, 2008, pp. 1322–1328.

[111] L. Breiman, "Random forests," *Machine learning*, pp. 5–32, 2001.

[112] D. Morawiec, "sklearn-porter," transpile trained scikit-learn estimators to C, Java, JavaScript and others. [Online]. Available: https://github.com/nok/sklearn-porter

[113] Uglifyjs is a javascript parser, minifier, compressor and beautifier toolkit. [Online]. Available: https://github.com/mishoo/UglifyJS

[114] Cnametracking uncloaker. [Online]. Available: https://chrome.google.com/webstore/detail/cnametracking-uncloaker/fhhdlfepbipknmeclodhcapbkmpdehkb

[115] J. Brookman, P. Rouge, A. Alva, and C. Yeung, "Cross-device tracking: Measurement and disclosures," *Proceedings on PET*, vol. 2017, no. 2, pp. 133–148, 2017.

[116] V. L. Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," *Proceedings of NDSS*, 2019.

[117] D. Zeber, S. Bird, C. Oliveira, W. Rudametkin, I. Segall, F. Wollsén, and M. Lopatka, "The representativeness of automated web crawls as a surrogate for human browsing," in *Proceedings of WWW*, 2020, pp. 167–178.

[118] L. Invernizzi, K. Thomas, A. Kapravelos, O. Comanescu, J.-M. Picod, and E. Bursztein, "Cloak of visibility: Detecting when machines browse a different web," in *Proceedings of IEEE S&P*, 2016, pp. 743–758.

[119] H. Jonker, B. Krumnow, and G. Vlot, "Fingerprint surface-based detection of web bot detectors," in *Proceedings of ESORICS*. Springer, 2019, pp. 586–605.

[120] A. Vastel, W. Rudametkin, R. Rouvoy, and X. Blanc, "Fp-crawlers: studying the resilience of browser fingerprinting to block crawlers," in *MADWeb'20-NDSS*, 2020.

[121] H. Dao, J. Mazel, and K. Fukuda, "Cname cloaking-based tracking on the web: Characterization, detection, and protection," *IEEE TNSM*, vol. 18, no. 3, pp. 3873–3888, 2021.

[122] (2020) Nextdns cname cloaking blocklist. [Online]. Available: https://github.com/nextdns/cname-cloaking-blocklist

[123] (2020) Cname-cloaked trackers. [Online]. Available: https://github.com/AdguardTeam/cname-trackers

[124] V. Mishra, P. Laperdrix, A. Vastel, W. Rudametkin, R. Rouvoy, and M. Lopatka, "Don' t count me out: On the relevance of ip address in the tracking ecosystem," in *Proceedings of The Web Conference 2020*, 2020, pp. 808–815.

[125] Uploading data. [Online]. Available: https://docs.liveramp.com/connect/en/uploading-data.html

[126] N. Steinfeld, " "i agree to the terms and conditions" :(how) do users read privacy policies online? an eye-tracking experiment," *Computers in human behavior*, vol. 55, pp. 992–1000, 2016.

[127] Adguard cname original trackers list. [Online]. Available: https://github.com/AdguardTeam/cname-trackers/blob/master/combined$_o$riginal$_t$rackers.txt

[128] A. Aliyeva and M. Egele, "Oversharing is not caring: How cname cloaking can expose your session cookies," in *Proceedings of ASIACCS*, 2021, pp. 1–12.

[129] T. Ren, A. Wittman, L. De Carli, and D. Davidson, "An analysis of first-party cookie exfiltration due to cname redirections," in *Proceedings of MADWeb*, 2021, pp. 1–11.

[130] Y. Dimova, G. Acar, L. Olejnik, W. Joosen, and T. Van Goethem, "The cname of the game: Large-scale analysis of dns-based tracking evasion," *Proceedings on PET*, vol. 2021, no. 3, pp. 393–411, 2021.

[131] Nextdns cname cloaking blocklist. [Online]. Available: https://github.com/nextdns/cname-cloaking-blocklist/blob/master/domains

[132] Cname cloaking and bounce tracking defense. [Online]. Available: https://webkit.org/blog/11338/cname-cloaking-and-bounce-tracking-defense/

[133] Characterizing cname cloaking-based tracking. [Online]. Available: https://blog.apnic.net/2020/08/04/characterizing-cname-cloaking-based-tracking/

[134] Firefox rolls out total cookie protection by default to all users worldwide. [Online]. Available: https://blog.mozilla.org/en/products/firefox/firefox-rolls-out-total-cookie-protection-by-default-to-all-users-worldwide/

[135] Manifest v3. [Online]. Available: https://developer.chrome.com/docs/extensions/mv3/

[136] M. Büchi, N. Just, and M. Latzer, "Caring is not enough: the importance of internet skills for online privacy protection," *Information, Communication & Society*, vol. 20, no. 8, pp. 1261–1278, 2017.

[137] (2021) The nsa and cia use ad blockers because online advertising is so dangerous. [Online]. Available: https://www.vice.com/en/article/93ypke/the-nsa-and-cia-use-ad-blockers-because-online-advertising-is-so-dangerous/

[138] O. Starov and N. Nikiforakis, "Privacymeter: Designing and developing a privacy-preserving browser extension," in *Proceedings of ESSoS*. Springer, 2018, pp. 77–95.

[139] Q. Chen, P. Snyder, B. Livshits, and A. Kapravelos, "Improving web content blocking with event-loop-turn granularity javascript signatures," *arXiv preprint arXiv:2005.11910*, 2020.

A

CNAME cloaking-based tracking appendices

As an alternative dataset, we also analyze the HAR (HTTP Archive) dataset [84] from 2016 Oct to 2020 Jan with the historic DNS data. This dataset was collected by test servers located in the US. They do not fully cover the Alexa list, so we only check the crawled sites that appeared in the Alexa list. Thus, one limitation of this data is that we cannot normalize the results (the number of analyzed sites are different in datasets). The details are shown in Table A.1.

Table A.1: Longitudinal HTTP Archive (HAR) datasets.

	2016 Oct	2017 Oct	2018 Oct	2019 Oct	2020 Jan
Alexa top sites	100K	100K	100K	100K	300K
Matched HAR sites	95,532	74,993	54,847	74,381	207,892
CNAME cloaking sites	1,005	1,122	1,012	1,025	1,584
Overlap sites	20,241	20,241	20,241	20,241	20,241
CNAME cloaking sites	537	603	681	699	795
Historical DNS coverage	68%	75%	90%	85%	85%

At a glance, the number of websites containing CNAME cloaking-based tracking is slightly stable in the Alexa top sites. However, we again conjecture that the number of CNAME cloaking sites increases over time in the overlap sites (20,241 sites). In summary, although the measurement environments affect detailed statistics, we confirm that two datasets (our dataset and HAR dataset) consistently demonstrate CNAME cloaking-based tracking occurrence.

B

PII Leakage-based tracking appendix

B.1 Supported hash functions and encodings for leak detection

Supported hashes, ecodings, and checksum: base16, base32, base32hex, base58, base64, gz, bzip2, deflate; and md2, md4, md5, sha1, sha224, sha256, sha384, sha512, crc16, crc32, sha3_224, sh3_256, sh3_384, sh3_512, ripemd_128, ripemd_169, ripemd_256, ripemd_320, whirlpool, rot13, snefru128, snefru256, adler32, blake2s, blake2b.

B.2 Performance analysis of machine learning model

The results obtained using the test set for requests linked to PII leakage on 20% of the dataset (preserved 13,315 for this test) are shown in Table Table B.2. We first show that our method detects requests related to PII leakage effectively. We achieve 0.72 of F1 score for leaked requests and 1.000 for non-leaked requests. By manually

analyzing some false negatives and some false positives, we find that requests linked to PII leakage have the same attributes as requests without it. For example, a request from *buyma.us* that leaks PII to Facebook.com via requests *https://www.facebook.com/tr/* by data payload of HTTP method POST. On the contrary, 140 requests that do not leak PII are predicted as false positive by the machine learning model. their request URL contains detailed tracking of user actions (including browser, device, and IP location) that make the length of this URL request is similar to request-related to PII leakage. However, they are also blocked by the Easy Privacy lists.

Table B.1: A grid of parameter settings for each algorithm of grid search optimization procedure.

Algorithm	**Parameter**	**Value**
Gradient Boosting	learning_rate	[0.1, 0.05, 0.01]
	loss	["deviance"]
	max_features	[0.3, 0.1]
	min_samples_leaf	[100,150],
	max_depth	[2,4, 6,8,10],
Decision Tree	criterion	['gini', 'entropy']
	min_samples_leaf	[1, 3, 10]
	min_samples_split	[2, 3, 10]
	max_depth =	[2,4,6,8,10]
AdaBoost	learning_rate	[0.1, 0.05, 0.01]
	algorithm	["SAMME","SAMME.R"]
	base_estimator__criterion	["gini", "entropy"]
	base_estimator__splitter	["best", "random"]
Random Forest	criterion	['gini', 'entropy']
	max_features	[1, 3, 10]
	min_samples_leaf	[1, 3, 10]
	min_samples_split	[2, 3, 10]
	bootstrap	[True, False]
Extra Trees	criterion	['gini', 'entropy']
	max_features	[1, 3, 10]
	min_samples_leaf	[1, 3, 10]
	min_samples_split	[2, 3, 10]
	bootstrap	[True, False]

Table B.2: Detection performance of machine learning approach

Class	Precision	Recall	F1 score
Non-PII leakage requests	1.000	0.990	1.000
PII leakage requests	0.640	0.840	0.720

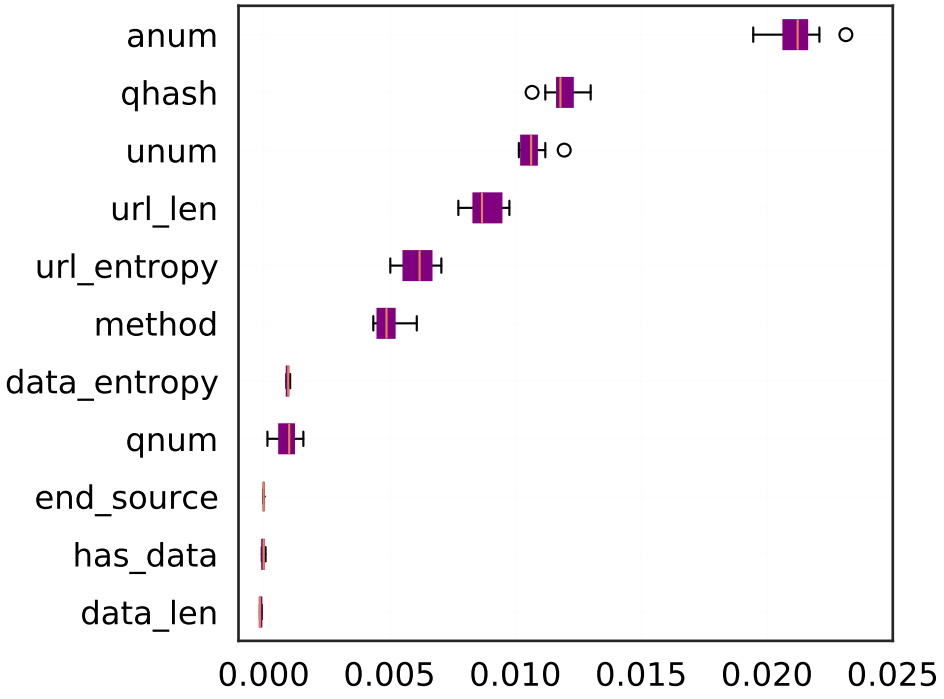


Figure B.1: Permutation importance of the selected model for PII leakage. The box extends from the lower to upper quartile values of the data, with a line at the median. The whiskers indicate variability outside the upper and lower quartiles. The number of times a feature is randomly shuffled is n_repeats = 10.

B.3 Permutation importance

To discover discriminative features for the detection, we investigate the permutation importance [111] to calculate the feature importance of the selected classifier for our dataset. Note that, larger values indicate higher importance.

Figure B.1 shows the feature permutation importance of the model for detecting the requests. The result reveals that the number of key-value pairs (*anum*) has the highest

importance. We assume that almost all leaked requests have at least one key-value pair containing a set of information sent to the third-party servers. Besides, the hash length appearing(*qhash*) and the number of number appearing (*unum*) are discriminative features for leaked request detection.