Net-track: Generic Web Tracking Detection Using Packet Metadata

Dongkeun Lee
dklee98@korea.ac.kr
Korea University
Seoul, Republic of Korea

Minwoo Joo
mw.joo@samsung.com
Samsung Research
Seoul, Republic of Korea

Wonjun Lee
wlee@korea.ac.kr
Korea University
Seoul, Republic of Korea

ABSTRACT

While third-party trackers breach users' privacy by compiling large amounts of personal data through web tracking techniques, combating these trackers is still left at the hand of each user. Although network operators may attempt a network-wide detection of trackers through inspecting all web traffic inside the network, their methods are not only privacy-intrusive but of limited accuracy as these are susceptible to domain changes or ineffective against encrypted traffic. To this end, in this paper, we propose *Net-track*, a novel approach to managing a secure web environment through platform-independent, encryption-agnostic detection of trackers. Utilizing only side-channel data from network traffic that are still available when encrypted, Net-track accurately detects trackers network-wide, irrespective of user's browsers or devices without looking into packet payloads or resources fetched from the web server. This prevents user data from leaking to tracking servers in a privacy-preserving manner. By measuring statistics from traffic traces and their similarities, we show distinctions between benign traffic and tracker traffic in their traffic patterns and build Net-track based on the features that fully capture trackers' distinctive characteristics. Evaluation results show that Net-track is able to detect trackers with 94.02% accuracy and can even discover new trackers yet unrecognized by existing filter lists. Furthermore, Net-track shows its potential for real-time detection, maintaining its performance when using only a portion of each traffic trace.

CCS CONCEPTS

• **Security and privacy** → *Network security*; • **Networks** → *Network privacy and anonymity*; *Network monitoring*.

KEYWORDS

encrypted traffic analysis, machine learning, security management, third-party tracker, web security

ACM Reference Format:
Dongkeun Lee, Minwoo Joo, and Wonjun Lee. 2023. Net-track: Generic Web Tracking Detection Using Packet Metadata. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3543507.3583372

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583372

1 INTRODUCTION

With the ever-increasing attention towards online privacy, protecting users' personal data has become a crucial issue in making a secure web environment. While policies such as the ePrivacy Directive (ePD) [32] or the EU General Data Protection Regulation (GDPR) [31] have been implemented as a response, there still remain factors threatening users' data privacy [6, 21]. One of those main threats are third-party trackers, commonly embedded in websites visited by users in the form of advertisements or web beacons.

Third-party trackers breach users' privacy by compiling large amounts of personal data through web tracking techniques. These trackers collect information such as the user's location or browsing history using cookies or device/browser fingerprinting. It has been studied that there exist 22 trackers per site on average, with more than 81,000 of them in total [13]. Along with the numerous efforts made by the research community to combat trackers, the industry now trends to integrate these privacy-protecting features into their products (e.g., Mozilla Firefox [22], Apple Safari [4], or Brave [5]).

Still, this daunting task of privacy protection is left at the hand of each user. A worried user may detect trackers loaded at his browser based on the requested URL or by analyzing HTML/JavaScript codes. Since URL-based approaches (e.g., [1, 14, 33, 37]) are easily evaded by changing domains or using the first-party domain as a proxy, recent approaches (e.g., [8, 17, 18]) extract features from changes in HTML structures or JavaScript attributes to detect tracking-related resources. However, these approaches cannot be deployed across diverse platforms or devices as they require an instrumented browser to perform their dynamic code analysis. This leads to fragmentation in security management as users should individually consider their settings to find a method viable for their own protection.

On the other hand, network operators in a security-critical organization (e.g., enterprise or military) may attempt a holistic approach that detects trackers by performing Domain Name System (DNS) filtering or Deep Packet Inspection (DPI) against all web traffic inside the network. Their methods are, however, limited in accuracy as DNS-based approaches (e.g., [2, 24]) are coarse-grained and subject to evasions equal to those of URL-based approaches. Moreover, extracting features from HTTP headers or payloads with DPI (e.g., [15, 28]) is not only privacy-intrusive but ineffective against encrypted traffic, making these approaches impractical in today's web environment where 79.8% of all websites use HTTPS as a default [34]. Overall, we are in a dilemma over putting the task of tracker detection in charge of each user for better performance or managing it as a whole for wider applicability.

To address this issue, we present *Net-track*, a novel approach to managing a secure web environment through platform-independent, encryption-agnostic detection of trackers. Utilizing

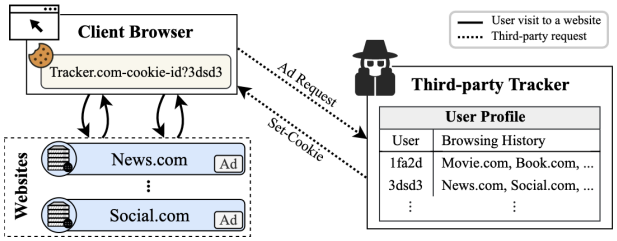

Figure 1: Simplified overview of a cookie-based tracker.

only side-channel data from network traffic that are still available when encrypted, Net-track accurately detects trackers network-wide, irrespective of user's browsers or devices but does not require inspecting packet payloads or resources fetched from the web server. This prevents user data from leaking to tracking servers in a privacy-preserving manner. Although there have been diverse studies on the potential of encrypted traffic analysis and its various usage (e.g., [3, 26, 30]), our work is the first attempt to apply this methodology in the field of tracker detection.

Our motivation stems from the observation that trackers' intrinsic functionalities (i.e., collecting and sending user data) generate distinctive traffic patterns that can be captured by analyzing their packet metadata: the length and direction of packets. By comparatively analyzing the client-server interaction for both benign traffic and tracker traffic, we elicit a set of features that make trackers distinguishable from benign traffic. The resulting three types of features: statistical features, box features, and sequential features, are incorporated into training a machine learning (ML) classifier to identify tracker traffic among the diverse traffic traces generated by users regardless of their browsers, applications, or devices.

We measure the statistics from traffic traces with their similarities to study the feasibility of utilizing side-channel data leaked from network traffic in detecting trackers. We test Net-track with diverse ML algorithms to evaluate its performance and further study its potential. Our results show that Net-track is able to detect trackers with 94.02% accuracy and can even discover new trackers yet unrecognized by existing filter lists. Besides, Net-track still maintains its performance when using only a portion of each traffic trace, showing its potential for real-time detection. The major contributions of this work are summarized as follows.

- To our best knowledge, we are the first to utilize only side-channel data in detecting trackers to achieve better user privacy that can be managed network-wide.
- We propose a platform-independent, encryption-agnostic solution to overcome the limitations of the previous approaches while mitigating the risk of privacy intrusion caused by inspecting packet payloads or resources.
- We provide an empirical analysis that shows the distinctions between benign traffic and tracker traffic in their traffic patterns and present a set of features that fully capture trackers' distinctive characteristics.
- We show that Net-track not only detects trackers with 94.02% accuracy but can discover new trackers that are not on the existing filter lists, maintaining its performance while using only the first few packets of each traffic trace.

2 BACKGROUND AND RELATED WORK

In this section, we briefly discuss the mechanisms of trackers' web tracking techniques with the previous studies on tracker detection and traffic classification.

2.1 Trackers: Types and Mechanisms

Third-party trackers utilize diverse tracking methods to identify each user and collect user data such as browsing history, location, or browser/device properties. *Stateful* tracking stores information on the user's device to recognize each user. Fig. 1 shows an overview of a cookie-based tracker, which is the most prevalent form of stateful tracking that collects users' browsing history through cross-site tracking. When the user first visits News.com, a third-party tracker, embedded in the website as an advertisement, sets a cookie in the user's browser. The cookie's value contains a unique identifier so that when the user later visits *Social.com*, the tracker can identify the user and notice that this user has visited *News.com* and *Social.com*.

Stateless tracking, on the other hand, does not require a tracker to store information on user devices. Instead, they collect device/user-specific information that can re-identify users when combined, such as OS/browser properties, user configuration, or browser extensions. Despite the difference in detail, these tracking methods all collect information about the user and send it to their tracking servers. This common functionality enables us to capture trackers' distinctive characteristics and utilize them in their identification. Details of analyzing the traffic patterns of trackers will be shown in Section 3.

2.2 Existing Studies on Tracker Detection

With the increasing attention to online privacy, many attempts have been made to combat trackers through their detection, leveraging diverse properties gained during the page load such as the structure and behavior of JavaScript code units or features of network requests. We note that our goal is not to depreciate the contributions of these approaches, but rather to suggest a different approach that can be used as a complement or as a viable alternative in today's web environment. We provide a thorough summary of the existing approaches to tracker detection in Appendix A.

(1) ***URL-based Detection:*** The most widely used methods adopted by privacy-sensitive users are content-blockers such as AdBlock Plus [1], uBlock Origin [33], and Ghostery [14]. These methods depend on manually curated filter lists, identifying tracking domains with URL patterns matching the predefined ruleset. While these methods are lightweight and easy to deploy, they are easily evaded by changing domains through domain generation algorithms (DGA) or proxies. Although Yu *et al.* [37] enabled a more robust detection by identifying suspicious data elements in third-party requests, their method still requires a tracker to use a consistent domain.

(2) ***JavaScript-based Detection:*** Wu *et al.* [36] and Ikram *et al.* [16] analyzed the static features of JavaScript code units to detect tracking JavaScript programs. As these are vulnerable to obfuscation techniques [20], recent approaches (e.g., [8, 17, 18]) perform dynamic code analysis to extract syntactic, semantic features from JavaScript executions. While these approaches are effective in detecting trackers and browser fingerprinting, they require a modification of Blink and V8 in the Chromium browser or an instrumented Firefox to log and attribute JavaScript behavior to document object

model (DOM) modifications and other network requests, limiting their applicability across diverse platforms. Net-track does not require user-level modifications for its real-world application and is complementary to these existing approaches. We can perform a more robust detection by combining them as a layered defense.

(3) ***Network-based Detection:*** Aside from the DNS-based approaches (e.g., [2, 24]), which simply perform DNS filtering and thus suffer from the same limitations of those URL-based approaches, several other approaches utilized features from HTTP traffic or the relationship between tracking domains. Gugelmann *et al.* [15] and Shuba *et al.* [28] proposed using a set of features obtained by inspecting HTTP headers and payloads (e.g., percentage of third-party requests or requests with cookies, domains in HTTP Referer). While their methods can detect trackers regardless of user's browsers or devices, applying them to encrypted traffic (e.g., through TLS proxies), which account for the majority of today's web traffic, may incur high overhead and threats to a man-in-the-middle attack. TrackSign [7], on the other hand, detects trackers by identifying code fragments shared across tracking domains. Although this can effectively discover new trackers through a network-wide analysis, it needs to fetch the entire resource to compute their code fingerprints. Net-track can be seen as a viable alternative to these approaches as it is encryption-agnostic and preserves its detection accuracy even when using only a part of each traffic trace.

2.3 Research Efforts on Traffic Classification

There have been various efforts to manage the security of the network by classifying network traffic with diverse side-channel data (e.g., [27, 38]). Anderson *et al.* [3] detected malware traffic based on packet length and TLS handshake metadata, while Taylor *et al.* [30] proposed identifying smartphone apps based on packet length and direction. Moreover, several work explored the feasibility of fingerprinting web pages within the same website using packet length information [26] or bursts in content distribution network (CDN) traffic [35]. There also has been a line of research utilizing deep learning in their classification. Sirinam *et al.* [29] studied fingerprinting websites against defended Tor traffic based on Convolutional Neural Network (CNN). On the other hand, Cui *et al.* [10] utilized Long Short-Term Memory (LSTM) as well as CNN to fingerprint websites with imperfect traffic traces, and Chen *et al.* [9] classified diverse types of traffic using LSTM with a sequence of message size as features. In this paper, we leverage only packet metadata: the length and direction of packets, in preserving users' privacy against tracker traffic inside the network.

3 ANALYSIS OF REAL-WORLD TRAFFIC

We begin by showing the results from studying the difference between benign traffic and tracker traffic in their traffic patterns. Our results show the feasibility of utilizing side-channel data from network traffic in distinguishing tracker traffic from benign traffic.

3.1 Dataset Collection

We first collect a real-world traffic dataset by visiting the homepages of the top-20k Alexa websites with Chrome browser using Selenium. We employ tshark scripts to capture all traffic generated during the page load for 120 seconds, in order to ensure that all flows are fully captured. Note that visiting a single website incurs multiple requests to different domains as there are various third parties engaged in the web page loading process (e.g., CDN, advertisement servers, or tracking servers). We then divide the resulting pcap file in terms of connection, i.e., make multiple pcap files each containing a single connection between the client and the domain. This makes us capture each client-server interaction with diverse third parties as well as with the host of the visiting website. The data collection is performed within a period of four weeks, resulting in a dataset consisting of about 350k traffic traces.

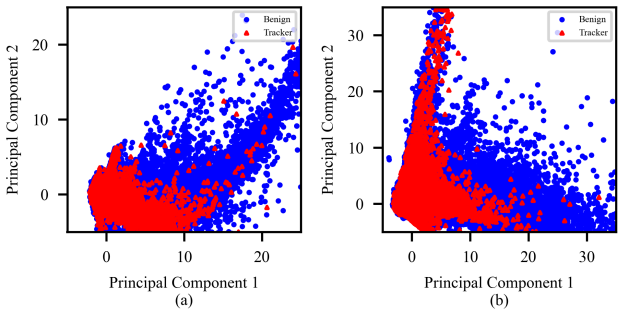

Figure 2: Principal component analysis results of statistical values. (a) Analysis of statistics from downlink packets, (b) Analysis of the entire statistics.

We now set the ground truth by labeling each trace as tracker or benign based on the requested URL. We use the two most popular open-source filter lists: EasyList [11] and EasyPrivacy [12], as our filter lists and label each trace based on whether it contains a request of which URL matches the rules on any of these filter lists, i.e., classified as tracker by filter lists. We note that while EasyList targets both advertisements and trackers, we do not discriminate between the two as most domain hosting advertisements also track users. It is also worth noting that despite the well-known limitations, filter lists are a reasonable source of ground truth when considering the time and labor required to build a more accurate, manually generated set of ground truth and are therefore still widely used in related work. Our final labeled dataset consists of 222,009 benign traffic traces and 126,664 tracker traffic traces.

3.2 Statistics from Traffic Traces

The first type of traffic features of our interest are the statistical values of packet length. As many applications show asymmetric statistical properties in the client-to-server (uplink) and server-to-client (downlink) directions, we divide the sequence of packet length from each traffic trace into three types of sequences: the length of uplink packets, the length of downlink packets, and the length of all packets in the flow. We combine 18 types of statistical values computed from each type of sequence, which are motivated by Taylor *et al.* [30], with 8 flow-level features that capture the characteristics of the entire flow, resulting in a total of 62 statistical features. The former 18 values are comprised of the minimum, maximum, mean, median absolute deviation (MAD), standard deviation (STD), variance, skew, kurtosis, percentiles (from 10% to 90%), and the number of elements in the sequence. Flow-level features

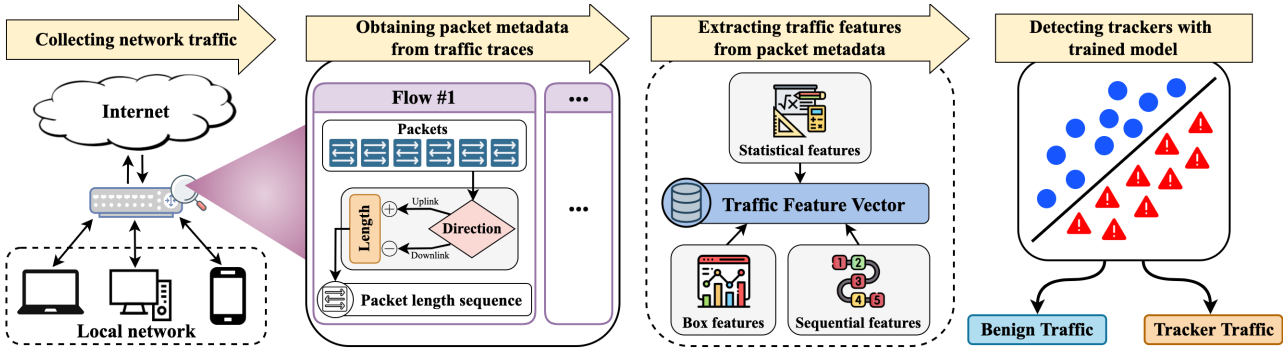

Figure 3: Net-track architecture: A workflow overview of network-wide tracker detection using packet metadata.

comprise the sum and the information entropy of packet length in each type of sequence, and the ratio of uplink packets to downlink packets in terms of their total length and count.

For the resulting set of statistical features, we perform principal component analysis (PCA) to study the overall distribution of their values. We reduce the multi-dimensional feature values into two principal components and visualize them as a scatter plot (Fig. 2). We can observe that for both Fig. 2a and Fig. 2b, traces of tracker traffic are more converged than benign traffic which shows their wider distribution. We attribute this difference to trackers performing similar functionalities, leading to more distinctive, shared patterns in their traces. Benign traffic, on the other hand, lacks commonalities compared to trackers as it is diverse in its types as well as its applications. We provide additional experimental results gained from analyzing real-world traffic traces in Appendix B.

4 DESIGN OF NET-TRACK

This section introduces the overall architecture of Net-track along with the details of our feature set and its selection process. The notations used in this paper are illustrated in Table 1.

4.1 System Architecture

The overall workflow of Net-track is depicted in Fig. 3. Net-track first collects real-world traffic traces by monitoring the network traffic inside its local network. We note that while we initially capture the entire traffic generated during the page load and separate them into multiple pcap files when performing our data collection, Net-track, when deployed real-world, can monitor every TCP connection to record the length of packets in each flow, generating an abstraction of those flows in the form of packet length sequence.

Net-track then extracts three types of features from the obtained packet length sequence: statistical features, box features, and sequential features, which fully represent the characteristics of tracker traffic. To further exploit the potential of these features in trackers' identification, we carry out a series of feature engineering (Section 4.3). The overall extraction process is summarized as Algorithm 1.

Finally, all these features are incorporated into training a classifier that distinguishes tracker traffic from benign traffic. We test diverse machine learning algorithms and deep learning models to make the best of Net-track's performance. The resulting classifier is applied to unknown traffic traces to label them as benign or tracker.

Algorithm 1: Feature Extraction Process of Net-track

Input: Sequence of packets $P = (p_1, ..., p_N)$ in the target flow
Output: Traffic feature vector V

Set empty lists as A, U, D, and S
forall $p_i \in P$ **do:**
  $l \leftarrow length(p_i)$
  $A.append(l)$
  if $is\_uplink\_packet(p_i)$ **then:**
    $U.append(l)$
    $S.append(l)$
  else:
    $D.append(l)$
    $S.append(-l)$
end
Calculate $STAT$ each for U, D and A
Calculate BOX for both U and D with bin size as 25 and maximum length as 1500
$BOX \leftarrow pca\_reduction(BOX, n\_components=20)$
$SEQ \leftarrow slice(S, begin=0, end=15)$
$V \leftarrow concat(STAT, BOX, SEQ)$
return V

4.2 Experimental Setup

To select the feature set that can fully exploit the potential of Net-track, we perform a series of preliminary evaluations. We study the effect of the changes in features on the performance of Net-track in terms of accuracy and training time. We choose the Random Forest (RF) classifier implemented in Scikit-learn [23] as our ML algorithm. To prevent the process of this feature selection from affecting the evaluation of Net-track's final performance, we split the entire dataset into 20% validation set and 80% test set and use only the validation set during the feature selection process. Further, to mitigate the impact of dataset partitioning, we apply 10-fold cross-validation over our validation set, dividing it into 10 equally-sized subsets and using one part as the testing set with the rest as the training set. After 10 rounds of iteration, we use the average as the final result. The machine used in this experiment has a 10-core Intel i9-7900X CPU, GeForce GT 1030 GPU, 32 GB RAM, 4 TB hard disk, and is installed with an Ubuntu 18.04 OS.

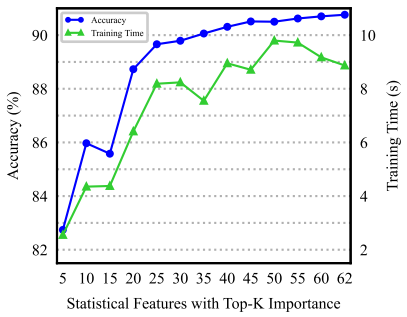

Figure 4: Effect of statistical features on the performance of Net-track.

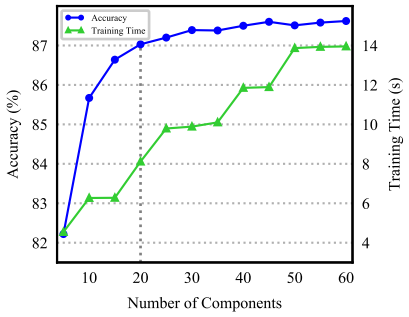

Figure 5: Effect of PCA on box features on the performance of Net-track.

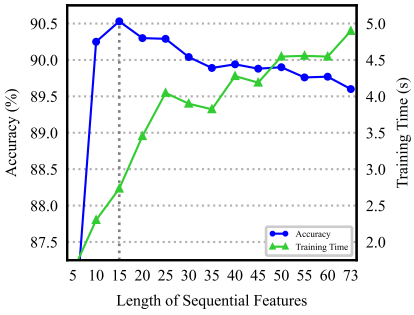

Figure 6: Effect of the length of sequential features on the performance of Net-track.

Table 1: List of Notations.

Notation	Meaning
A	Packet length sequence of all packets
U	Packet length sequence of uplink packets
D	Packet length sequence of downlink packets
S	Signed sequence of packet length
STAT	Feature vector of statistical features
BOX	Feature vector of box features
SEQ	Feature vector of sequential features

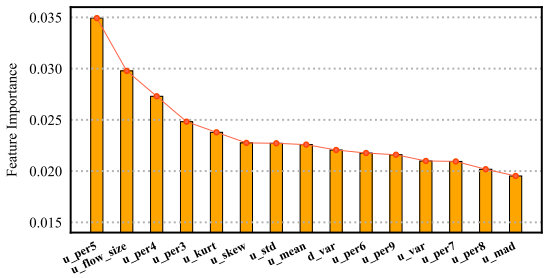

Figure 7: Top 15 statistical features that contribute most to the detection of trackers on our dataset.

4.3 Selecting the Feature Set

We now present the features that Net-track utilizes in its detection. Based on the original packet length sequence (*A*), we build three additional sequences: *U*, *D*, and *S*, and extract our features from these sequences. For *S*, its sign indicates the direction of the packet: positive for uplink and negative for downlink. We explain more about our design choices in Appendix C.

(1) ***Statistical Features:*** The first type of features, statistical features, are computed from *U*, *D*, and *A* as discussed in Section 3.2, extracting 18 features from each sequence with 8 flow-level features from the entire flow. We quantify the contribution of each feature with feature importance gained when building an RF classifier based on Gini impurity. Fig. 7 shows the top 15 features that contribute most to identifying trackers in our dataset. The prefix 'u_' and 'd_' each denotes the features computed from *U* and *D*, respectively. 'per*K*' and 'flow_size' each denotes the $K0^{th}$ percentile and the sum of packet length in the sequence. We can see that several features contribute more than others, although not greatly.

Motivated by the result, we initially tried using certain features with their contribution over a certain threshold, expecting to reduce the feature dimension while preserving the accuracy. Fig. 4 shows the result when using statistical features with top-k importance. Surprisingly, we observe that while the classification accuracy gradually increases, this does not apply to the training time. Using the entire 62 statistical features showed the highest accuracy with its training time similar to or even lower than the case of using a smaller number of features. According to the result, we add all statistical features to our feature set.

(2) ***Box Features:*** We then extract box features, which are the distribution of packet length counted in terms of each bin. For each traffic trace, each uplink (downlink) packet in the flow is classified into equally sized bins with a size of 25 according to its length. A packet longer than 1500 bytes is counted as the 61st bin. This results in a 61-dimensional feature vector for each *U* and *D*, a total of a 122-dimension. As box features are initially too sparse to be used in their original form, we perform PCA with a different number of principal components to reduce their dimensionality.

In Fig. 5, Using *N* components indicates reducing the box features into a *N*-dimensional vector. We can see that while increasing the number of components leads to higher accuracy, it takes nearly double the training time to get an enhancement of less than 1%p. This is because increasing the number of components does not necessarily lead to an accurate classification as the variance of the dataset is concentrated only on the first few components. A latecomer may have a negligible effect, i.e., contribution, on the detection of trackers. Based on the result, we use 20 components as it shows moderate accuracy with a shorter training time.

(3) ***Sequential Features:*** The last type of features are the sequential features, which denote the signed sequence of packet length

Table 2: Hyperparameters of ML Algorithms.

Classifiers	Hyperparameters
DT	*criterion*='entropy', *max_depth*=None, *max_features*=None
k-NN	*n_neighbors*=1, *metric*='minkowski', *p*=1
MLP	*hidden_layer_sizes*=[256,256,256], *activation*='relu'
RF	*criterion*='gini', *max_depth*=None, *max_features*='sqrt', *n_estimators*=55

(*S*). The main factor of our consideration is their length, i.e., the length of *S*. Starting from 5, we increase the length with a step size of 5 until 60 and then 73, which is the 95^{th} percentile of the length of all *S* in our dataset. As packet length sequences are in arbitrary lengths, we pad shorter sequences by appending zeros to them and truncate the longer ones. The results of our experiment are shown in Fig. 6. The accuracy and training time when using a packet sequence of length 5 are 76.11% and 0.9989 s, respectively.

We observe that while the accuracy increases until the length of sequential features becomes 15, it continues to drop when it becomes longer, albeit with a longer training time. We attribute this result to the length of traces, of which the median is shorter than 20 for both benign traffic and tracker traffic. Also, for the average length of traces, benign traffic and tracker traffic each show an average of 29.32 and 18.31. Note that traces of tracker traffic are generally shorter as they convey less content than benign traffic. When the length of sequential features becomes longer, there comes more *S* padded with zeros. This negatively impacts the classification as there are more null-valued features in the feature set. According to our results, we set the length of sequential features to 15.

5 PERFORMANCE EVALUATION

This section presents the results of our evaluation, which is designed to answer the following four key research questions: (1) how accurate is Net-track in detecting trackers with only side-channel data? (2) how effective is Net-track in discovering trackers that are not on the filter lists? (3) how well can Net-track preserve its performance when using only a part of each traffic trace? and (4) can deep learning enhance the performance of Net-track?

Our evaluation is performed in the same setting as discussed in Section 4.2, but by applying 10-fold cross-validation on the test set. We compare our predictions made by Net-track with the labels derived from the filter lists described in Section 3.1. We then evaluate how accurately Net-track can reproduce those labels and perform a manual analysis on cases where those results collide, finding that Net-track can identify many new trackers missed by existing filter lists. Our ML algorithms are implemented in Scikit-learn [23] and the deep learning models are implemented in Python using Keras with Tensorflow as the back-end [19]. We perform grid search over the validation set to tune the hyperparameters. For the run time, the inference time is averaged by the number of traces in the testing set. We note that each testing set is a 10% subset of the entire test set as we perform 10-fold cross-validation in our evaluation.

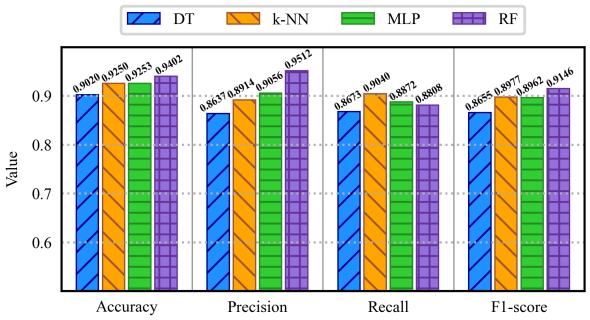

Figure 8: Detection performance of Net-track with different ML algorithms.

Table 3: Run time of Net-track with different ML algorithms.

	DT	*k*-NN	MLP	RF
Training Time (s)	27.7080	0.0516	1638.53	73.3319
Inference Time (ms)	0.0011	12.813	0.0209	0.0163

5.1 Detection Performance of Net-track

Net-track incorporates the extracted features into training an ML classifier to perform its detection. We choose four ML algorithms that are widely used in related work: Decision Tree (DT), *k*-Nearest Neighbors (*k*-NN), Multi-layer Perceptron (MLP), and Random Forest (RF), and comparatively analyze their performance. Table 2 shows the details of our hyperparameters selection. Along with the accuracy used in Section 4.3, we use precision, recall, and F1-score as our metrics, which are defined as:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad (1)$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (2)$$

$$F1\text{-}score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (3)$$

The overall results of our experiment are shown in Fig. 8 and Table 3. We can observe that while DT is faster than other ML algorithms (except for *k*-NN that simply 'remembers' the training data in its training stage), its detection performance is the worst, with 90.2% accuracy and 86.55% F1-score. On the other hand, *k*-NN and MLP show a somewhat opposite result, although with similar levels of performance with about 92.5% accuracy. While *k*-NN is the slowest in making its inference, MLP takes the longest to train its model. RF is the highest both in its accuracy (94.02%) and precision (95.12%), accompanied by a moderate training/inference time. As RF consists of multiple DTs and makes its final prediction based on the probability estimate across the trees, it is robust to overfitting and thus results in better performance. We also note that we can further reduce the run time of RF and *k*-NN through multiprocessing. When we set *n_jobs* to 10, the inference time is reduced to 7.209 ms and 0.0054 ms for *k*-NN and RF, respectively, while maintaining their detection performance.

Table 4: Results of manual analysis on cases which Net-track classified as tracker while filter lists labeled as benign.

Traffic Type		#
Tracker	Type 1	38 (19 %)
	Type 2	15 (7.5 %)
	Type 3	16 (8 %)
Benign		131 (65.5 %)

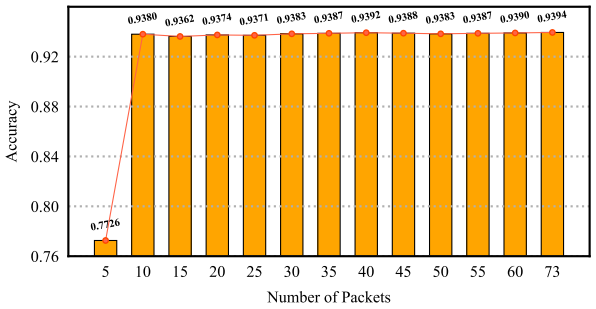

Figure 9: Detection performance of Net-track when using only the first *n* packets.

The above results show that Net-track has high precision at 95% in detecting trackers' privacy-intrusive behaviors. With Net-track's high precision, we can perform pinpoint blockage of trackers without impeding the benign. Filter lists fail to do so due to their over-blocking of resources, blocking entire domains or URLs in a given ruleset. Iqbal *et al.* [18] found out that resources of the same type (e.g., scripts) fetched from the same domain may show different functionalities depending on the context they are requested. Such cases are recklessly blocked by filter lists and increase the number of 'False Negative', i.e., the number of cases filter lists labeled as tracker while Net-track classified as benign, making Net-track's recall relatively lower than its precision.

What also makes these results worthwhile is that Net-track's performance is attained without analyzing resources loaded at the application layer nor inspecting contents in the HTTP payloads. By leveraging only side-channel data from network traffic, Net-track achieves a level of accuracy comparable to other approaches. We will discuss more on our future aim to develop Net-track in Section 6. Based on the overall results, we choose RF as our classifier. We note that the results in the following Section 5.2 and Section 5.3 are obtained using our RF classifier.

5.2 Discovering New Trackers

In this section, we show how well Net-track can discover new trackers based on its learning and can further address filter lists' limitations. As filter lists consist of a set of crowdsourced rules generated by human experts, which are the results of their manual analysis on millions of websites, it is hard for those filter lists to identify new tracking services or react to evasions (e.g., domain variants) in a timely manner. Net-track gives a hand with this arduous struggle for privacy.

We perform a case study on 200 samples of randomly selected traffic traces that Net-track classified as tracker while labeled as benign by filter lists. After analyzing their client-server interaction and the fetched resources, we label each traffic trace as tracker traffic if it corresponds to any of the following criteria:

- **Type 1:** This type of traffic fetches advertisements (ads) from their ad servers. As stated in Section 3.1, there is no clear distinction between ads and tracking resources as most domain hosting ads also track users by identifying users through cookies or sending *uid* with their requests.
- **Type 2:** This type of traffic downloads tracking-related JavaScript files, which directly set cookies in browsers or perform device fingerprinting to collect user information.
- **Type 3:** Initiated by tracking pixels or tracking scripts, this type of traffic communicates with third-party tracking services to send information to their servers. For example, sending the user's browsing history, location, or measurements gained using browser API in the form of parameters or payloads of HTTP GET/POST methods.

We note that as many of these tracker traffic show 'mixed' features of the above, it is hard to strictly group them in each category. Therefore, we only consider their main functionality and show the distribution of tracker traffic according to their type. Table 4 shows the details of our analysis results.

Our results demonstrate that Net-track can effectively detect trackers that are not on the existing filter lists. This also further reinforces Net-track's precision (discussed in Section 5.1) as a considerable portion (34.5%) of those 'detection errors' were indeed trackers that have not yet been discovered. By analyzing these newly detected trackers and the existing filter lists, we find that EasyList applies to *mc.yandex.ru* but not to *mc.yandex.com*. It also blocks *adtarget.me* but not *adtarget.com.tr*, as these filter lists are subject to domain changes. Net-track also identifies tracking behaviors missed by existing filter lists such as downloading tracking-related script (conversations-embed.js) from *usemessages.com* or communicating with third-party tracking services, sending information to subdomains of *drift.com* (e.g., *targeting.api*, *metrics.api*) or sharing information between tracking services, i.e., cookie syncing, through *x.dlx.addthis.com*. It also detects a script (afterpay-1.x.js) on *afterpay.com*. Although *afterpay.com* is not essentially a tracking domain, its script collects information such as city, country, device manufacturer/model, and OS name/version at the user's browser.

Besides, when investigating the newest version of the filter lists at the time of this writing, we observe that 37.68% of these newly found trackers are still unenrolled. We note that we use filter lists that are up to date at the moment of data collection in setting the ground truth and training our model, and it has passed more than ten months since then. Although given enough time, these manually curated filter lists still fail to adapt to changes in trackers and their evasions, as they are too slow to keep pace with the fast-changing nature of third-party trackers. With Net-track, we can provide these filter lists with a list of candidates that Net-track marked as suspicious, significantly reducing their scope of manual analysis and help complement these filter lists much more efficiently.

Table 5: Hyperparameters selection for LSTM.

Hyperparameters	Search Range	Final
Input Units	[50, ..., 300]	150
Hidden Layer Units	[64, ..., 256]	64
Activation	[tanh, relu]	tanh
Dropout	[0.1, ..., 0.5]	0.3
Batch Size	[32, ..., 512]	256
Training Epochs	[10, ..., 100]	50

5.3 Potential for Real-time Detection

In our previous experiments, we extract features from all packets in each traffic trace. But when assuming a more realistic environment where Net-track performs real-time classification of network traffic, it should be able to detect trackers in the midst of the connection, i.e., use as few packets as possible in making its decision. Therefore, to study Net-track's potential for real-time detection, we train our model with the features extracted only from the first n packets of each traffic trace and evaluate its performance.

Fig. 9 shows the details of the results. We can observe that except for the case of using the first 5 packets, Net-track maintains its accuracy of over 93%. It shows the lowest accuracy at 93.62% when using the first 15 packets and the highest accuracy at 93.94% when using the first 73 packets of each traffic trace. We note that the average length of tracker traffic in our dataset is 18.31. While both benign and tracker traffic show similar behavior patterns in the early stage of connection (first 5 packets), Net-track can extract informative features with a minimum of 10 packets as most trackers are amidst connection. Our results demonstrate that Net-track can perform its accurate detection even with insufficient data, showing its ability to identify tracker traffic within a short time after it is generated, instead of waiting for each flow to complete.

5.4 Net-track with Deep Learning

This section further explores the benefit deep learning might bring to the performance of Net-track. Among the various deep learning models, we choose LSTM, a specific type of recurrent neural network that specializes in classifying time-series data such as speech, video, or network traffic and is thus widely used in related work.

Table 5 shows the list and the values of the hyperparameters for our LSTM model. An input instance for our LSTM-based classifier is the signed sequence of packet length (S) from each traffic trace. For the length of each sequence, we use 15 (length of sequential features), 25 (average length of traces), and 73 (95^{th} percentile in the length of traces). Table 6 shows the details of our evaluation results. We can observe that our LSTM-based classifier performs similarly to other ML-based classifiers, but with a longer run time. To be more specific, while LSTM shows similar accuracy (93.50% when using a sequence of length 25) with our RF-based classifier (94.02%), it takes 11 times longer than RF in making its inference. Our results show that although deep learning may also lead to a comparable performance to other approaches (with a proper model and a careful selection of its hyperparameters), its intrinsic shortcomings in time complexity make it less suitable for making a timely decision.

Table 6: Detection performance of Net-track with LSTM.

Sequence Length	Accuracy	Precision	Recall	Train (s)	Inference (ms)
15	0.9287	0.9196	0.8808	771.03	0.1701
25	0.9350	0.9262	0.8922	923.80	0.1783
73	0.9349	0.9309	0.8867	1969.83	0.2274

6 DISCUSSION

For further study, we aim to apply Net-track as a back-end system in a layered architecture, acting as a source of information that feeds other systems. For example, network managers in an enterprise network can initially filter out tracker traffic in their local network using Net-track. When Net-track identifies tracker traffic from unknown sources (whether in real-time or using the entire flow), it can update firewall rules or tracking domain lists to block subsequent flows, protecting other users of the network. Users who are concerned more about their privacy may then equip additional browser-based defenses (e.g., [1, 14, 33]).

Also, when considering its real-world application, Net-track may be affected by changes in traffic characteristics. For example, as Net-track utilizes distinct properties that tracker traffic has regarding its packet length sequence, its features may be compromised by obfuscation in traffic patterns induced by dummy packets or changes in packet length. Addressing this challenge will be explored in our future work.

We can further enhance Net-track's performance by retraining its classifier. The accuracy of the filter lists greatly affects Net-track's performance as Net-track uses them to label its training data as tracker or benign. As discussed in Section 5.2, Net-track can help complement these filter lists. This in turn improves Net-track as we can retrain Net-track by re-labeling its training data with the enhanced filter lists. In fact, when we use the newest filter lists at the time of writing in setting the ground truth and retrain our model, we observe that our RF classifier shows a 4%p increase in recall while maintaining its accuracy and precision.

7 CONCLUSION

In this paper, we proposed a novel approach to tracker detection called Net-track, which can manage a secure, tracking-free web by utilizing only side-channel data leaked from network traffic in detecting trackers network-wide, independently of the user's browsers or devices. Our method is encryption-agnostic and does not inspect contents in packet payloads or resources loaded at the application layer, preventing leakage of user data to tracking servers in a privacy-preserving manner. The experimental results demonstrated that Net-track can identify tracker traffic among the diverse traffic traces with 94.02% accuracy and can also discover many new trackers that are not recognized by existing filter lists. Moreover, Net-track showed its potential for detecting trackers in real-time, preserving its accuracy while using only the first n packets of each traffic trace. Our future work aims to incorporate Net-track into a back-end system that can block trackers in real-time based on Net-track's discovery of new, unknown trackers.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable feedback. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (Ministry of Science & Information and Communication Technology) (No. 2019R1A2C2088812).

REFERENCES

[1] AdBlock Plus. Accessed: Sep. 18, 2022. https://adblockplus.org/
[2] AdGuard DNS. Accessed: Sep. 18, 2022. https://adguard-dns.com/
[3] Blake Anderson and David McGrew. Aug. 2017. Machine Learning for Encrypted Malware Traffic Classification: Accounting for Noisy Labels and Non-Stationarity. In *Proc. ACM KDD*. Halifax, NS, Canada, 1723–1732.
[4] Apple. Accessed: Sep. 17, 2022. Tracking Prevention in WebKit. WebKit. https://webkit.org/tracking-prevention/
[5] Brave. Accessed: Sep. 17, 2022. https://brave.com/
[6] Luca Bufalieri, Massimo La Morgia, Alessandro Mei, and Julinda Stefa. Oct. 2020. GDPR: When the Right to Access Personal Data Becomes a Threat. In *Proc. IEEE ICWS*. Virtual Event, 75–83.
[7] Ismael Castell-Uroz, Josep Solé-Pareta, and Pere Barlet-Ros. May 2021. TrackSign: Guided Web Tracking Discovery. In *Proc. IEEE INFOCOM*. Virtual Event, 1–10.
[8] Quan Chen, Peter Snyder, Ben Livshits, and Alexandros Kapravelos. May 2021. Detecting Filter List Evasion with Event-Loop-Turn Granularity JavaScript Signatures. In *Proc. IEEE S&P*. Virtual Event, 1715–1729.
[9] Wenxiong Chen, Feng Lyu, Fan Wu, Peng Yang, Guangtao Xue, and Minglu Li. 2021. Sequential Message Characterization for Early Classification of Encrypted Internet Traffic. *IEEE Trans. Veh. Technol.* 70, 4 (2021), 3746–3760.
[10] Weiqi Cui, Tao Chen, and Eric Chan-Tin. Dec. 2020. More Realistic Website Fingerprinting Using Deep Learning. In *Proc. IEEE ICDCS*. Singapore, 333–343.
[11] EasyList. Accessed: Sep. 21, 2022. https://easylist.to/easylist/easylist.txt
[12] EasyPrivacy. Accessed: Sep. 21, 2022. https://easylist.to/easylist/easyprivacy.txt
[13] Steven Englehardt and Arvind Narayanan. Oct. 2016. Online Tracking: A 1-Million-Site Measurement and Analysis. In *Proc. ACM CCS*. Vienna, Austria, 1388–1401.
[14] Ghostery. Accessed: Sep. 18, 2022. https://www.ghostery.com/
[15] David Gugelmann, Markus Happe, Bernhard Ager, and Vincent Lenders. Jun. 2015. An Automated Approach for Complementing Ad Blockers' Blacklists. In *Proc. PETS*. Philadelphia, PA, USA, 282–298.
[16] Muhammad Ikram, Hassan Asghar, Mohamed Ali Kaafar, Balachander Krishnamurthy, and Anirban Mahanti. Jan. 2017. Towards Seamless Tracking-Free Web: Improved Detection of Trackers via One-class Learning. In *Proc. PETS*. Minneapolis, MN, USA, 79–99.
[17] Umar Iqbal, Steven Englehardt, and Zubair Shafiq. May 2021. Fingerprinting the Fingerprinters: Learning to Detect Browser Fingerprinting Behaviors. In *Proc. IEEE S&P*. Virtual Event, 1143–1161.
[18] Umar Iqbal, Peter Snyder, Shitong Zhu, Benjamin Livshits, Zhiyun Qian, and Zubair Shafiq. May 2020. AdGraph: A Graph-Based Approach to Ad and Tracker Blocking. In *Proc. IEEE S&P*. Virtual Event, 763–776.
[19] Keras: the Python deep learning API. Accessed: Sep. 23, 2022. https://keras.io/
[20] Hoan Le, Federico Fallace, and Pere Barlet-Ros. Sep. 2017. Towards accurate detection of obfuscated web tracking. In *Proc. IEEE M&N*. Naples, Italy, 1–6.
[21] Célestin Matte, Nataliia Bielova, and Cristiana Santos. May 2020. Do Cookie Banners Respect my Choice?: Measuring Legal Compliance of Banners from IAB Europe's Transparency and Consent Framework. In *Proc. IEEE S&P*. Virtual Event, 791–809.
[22] Mozilla. Accessed: Sep. 17, 2022. Tracking Protection. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/Privacy/Tracking_Protection
[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* 12 (2011), 2825–2830.
[24] Pi-hole. Accessed: Sep. 18, 2022. https://pi-hole.net/
[25] Stan Salvador and Philip Chan. Oct. 2007. FastDTW: Toward accurate dynamic time warping in linear time and space. *Intell. Data Anal.* 11, 5 (Oct. 2007), 561–580.
[26] Meng Shen, Yiting Liu, Liehuang Zhu, Xiaojiang Du, and Jiankun Hu. 2021. Fine-Grained Webpage Fingerprinting Using Only Packet Length Information of Encrypted Traffic. *IEEE Trans. Inf. Forensics Secur.* 16 (2021), 2046–2059.
[27] Meng Shen, Yiting Liu, Liehuang Zhu, Ke Xu, Xiaojiang Du, and Nadra Guizani. 2020. Optimizing Feature Selection for Efficient Encrypted Traffic Classification: A Systematic Approach. *IEEE Network* 34, 4 (2020), 20–27.
[28] Anastasia Shuba, Athina Markopoulou, and Zubair Shafiq. Jul. 2018. NoMoAds: Effective and Efficient Cross-App Mobile Ad-Blocking. In *Proc. PETS*. Barcelona, Spain, 125–140.
[29] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. Oct. 2018. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In *Proc. ACM CCS*. Toronto, Canada, 1928–1943.
[30] Vincent F. Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. 2018. Robust Smartphone App Identification via Encrypted Network Traffic Analysis. *IEEE Trans. Inf. Forensics Secur.* 13, 1 (2018), 63–78.
[31] The European Parliament and the Council of the European Union. Apr. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union* (Apr. 2016).
[32] The European Parliament and the Council of the European Union. Jul. 2002. Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications). *Official Journal of the European Union* (Jul. 2002).
[33] uBlock Origin. Accessed: Sep. 18, 2022. https://github.com/gorhill/uBlock
[34] W3Techs. Accessed: Sep. 23, 2022. Usage statistics of Default protocol https for websites. https://w3techs.com/technologies/details/ce-httpsdefault
[35] Kailong Wang, Junzhe Zhang, Guangdong Bai, Ryan Ko, and Jin Song Dong. Apr. 2021. It's Not Just the Site, It's the Contents: Intra-Domain Fingerprinting Social Media Websites Through CDN Bursts. In *Proc. WWW*. Ljubljana, Slovenia, 2142–2153.
[36] Qianru Wu, Qixu Liu, Yuqing Zhang, Peng Liu, and Guanxing Wen. Sep. 2016. A Machine Learning Approach for Detecting Third-Party Trackers on the Web. In *Proc. ESORICS*. Heraklion, Greece, 238–258.
[37] Zhonghao Yu, Sam Macbeth, Konark Modi, and Josep M. Pujol. Apr. 2016. Tracking the Trackers. In *Proc. WWW*. Montréal, Québec, Canada, 121–132.
[38] Jingjing Zhao, Xuyang Jing, Zheng Yan, and Witold Pedrycz. 2021. Network traffic classification for data fusion: A survey. *Information Fusion* 72 (2021), 22–47.

A SUMMARY OF THE EXISTING WORK

Table 7 shows the strengths and weaknesses of existing work along with the ones of Net-track. Although there have been diverse approaches each leveraging its distinct properties, we note that there is no 'silver bullet' in the field of tracker detection, as they each suffer from their own limitations. By leveraging only packet metadata, Net-track is complementary to these approaches and can work as a basis of privacy protection that operates at the back end, enabling more precise detection of trackers when combined.

B ANALYZING FEATURES FROM PACKET LENGTH

We further study the distinctions between benign traffic and tracker traffic in terms of packet length, analyzing their distribution and measuring the similarity between those packet length sequences.

B.1 Distribution of Packet Length

To analyze the difference in the distribution of packet length between benign traffic and tracker traffic, we keep a count of each bin to which the length of each packet in the flow belongs, with the bin size as 25 and the maximum length as 1500 (e.g., any packet with length in the range [0,25) is counted as the first bin, [25,50) counted as the second). By normalizing the count of each bin with the total number of uplink (downlink) packets in the flow, we get the distribution of packet length for each traffic trace. For each bin, we compute the mean of its normalized count between all traces belonging to the same traffic type. The results are shown in Fig. 10.

We can observe that for both uplink packets (Fig. 10a) and downlink packets (Fig. 10b), the distribution of packet length differs according to the traffic type. What is noteworthy is that similar to the results in Fig. 2a, the distribution of downlink packets

Table 7: Summary of the existing work on tracker detection.

Properties	Ref.	Year	Advantages	Disadvantages
URL Request	[1, 14, 33]	2022	- Lightweight and easy to deploy - Support real-time detection	- Susceptible to domain changes - Require specific browser to install[1, 14, 33]
	[37]	2016		
DNS Query	[2, 24]	2022	- Does not require installation on each device - Support real-time detection	- Susceptible to domain changes - Does not work against encrypted DNS
Code Structure	[36]	2016	- Robust to domain changes - Does not require specific browser to deploy	- Susceptible to code obfuscation - Unable to detect in real-time
	[16]	2017		
JavaScript Behavior	[18]	2020	- Robust to domain changes and code obfuscation - Support real-time detection[18]	- Depend on specific browser functions - Require modifying the base browser to deploy
	[8, 17]	2021		
HTTP Header/Payload	[15]	2015	- Does not require installation on each device - Support real-time detection[28]	- Inspecting payloads may raise a privacy issue - Does not work against encrypted traffic
	[28]	2018		
Code Fingerprint	[7]	2021	- Robust to domain changes - Does not require specific browser to deploy	- Require downloading the entire file - Unable to detect in real-time
Packet Metadata	This Work	2023	- Platform-independent and encryption-agnostic - Can detect trackers with partial traces	- May be affected by changes in traffic characteristics (e.g., packet size randomization)

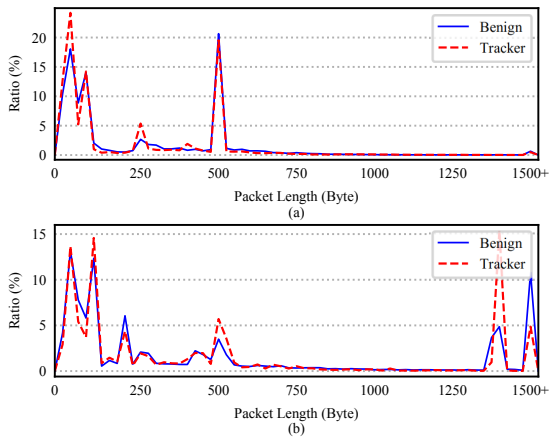

Figure 10: Distribution ratio of packet length. Packets are aggregated with their length into an interval size of 25. (a) Uplink packets, (b) Downlink packets.

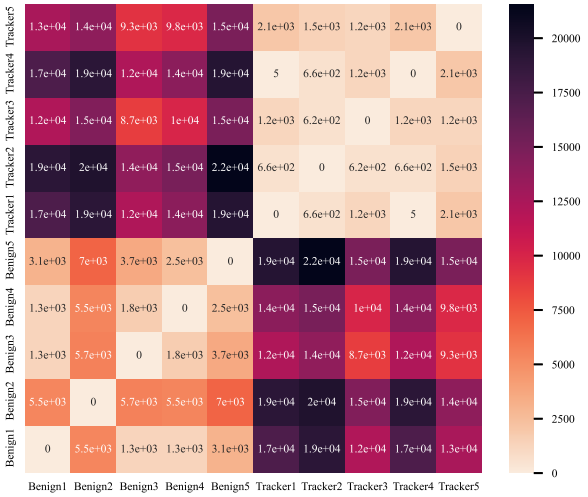

Figure 11: Similarity between packet length sequence of traffic traces measured in terms of DTW distance. A closer distance indicates higher similarity.

shows a greater difference than that of uplink packets. This is because trackers focus more on sending information to their tracking servers rather than downloading resources, while most benign traffic fetches web contents or other resources during the page load, generating downlink packets with larger payloads. Thus, the length of downlink packets leads to a more noticeable distinction between benign traffic and tracker traffic.

B.2 Similarity between Packet Sequences

In addition to the three types of sequences discussed in Section 3.2, we build a signed sequence of packet length, of which the sign indicates the direction of the packet: positive for uplink and negative for downlink. We then measure the similarity between traffic traces in terms of the distance between those signed sequences. As traffic traces are with an arbitrary length, we use dynamic time warping (DTW) [25] to measure their distance. Fig. 11 shows the similarity between traces that are randomly selected each from benign traffic and tracker traffic. We can observe that traces belonging to the same type show a higher similarity between each other while showing a

larger distance with traces in another type. It is worth noting that the similarity between tracker traffic is relatively more robust than that of benign traffic. As with the result in Section 3.2, traces of tracker traffic continue to show a convergence of their features, indicating the presence of their commonalities in patterns that make them distinguishable from benign traffic.

C EXPLORING CANDIDATE FEATURES

Feature engineering of Net-track involved examining myriads of approaches [27, 38] and measuring their effectiveness. We initially study diverse features that can be obtained from packet metadata. These include counting the number of uplink/downlink packet blocks (i.e., when packets are consecutively transmitted to the same direction) or accumulating packet length of the original packet length sequence for sequence abstraction. We also investigate relational features such as the magnitude, radius, covariance, and correlation coefficient of packet length in each trace. However, the effect of these features on the performance of Net-track falls short of our expectations, leading to our final feature set that can fully capture the distinctive characteristics of tracker traffic: statistical, box, and sequential features.

We further refine the resulting feature set by complementing our dataset with neighborhood-based collaborative filtering, which, however, only brings marginal enhancements. When the number of packets is small, there exist cases where some feature values are unavailable. For each missing feature value, we select three traces based on their similarity with the target trace and use their average as a prediction. Using this augmented dataset only brings negligible impact on Net-track's detection. Therefore, we choose to use our dataset in its original form.