# Transformer-based Model for Multi-tab Website Fingerprinting Attack

Zhaoxin Jin
School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications
Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education
Beijing, China
jzx3990@bupt.edu.cn

Tianbo Lu*
School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications
Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education
Beijing, China
lutb@bupt.edu.cn

Shuang Luo
School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications
Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education
Beijing, China
lok@bupt.edu.cn

Jiaze Shang
School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications
Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education
Beijing, China
sjz@bupt.edu.cn

## ABSTRACT

While the anonymous communication system Tor can protect user privacy, website fingerprinting (WF) attackers can still identify the websites that users access over encrypted network connections by analyzing the metadata generated during network communication. Despite the emergence of new WF attack techniques in recent years, most research in this area has focused on pure traffic traces generated from single-tab browsing behavior. However, multi-tab browsing behavior significantly degrades the performance of WF classification models based on the single-tab assumption. As a result, some research has shifted its focus to multi-tab WF attacks, although most of these works have limited utilization of the mixed information contained in multi-tab traces. In this paper, we propose an end-to-end multi-tab WF attack model, called **T**ransformer-based model for **M**ulti-tab **W**ebsite **F**ingerprinting attack (TMWF). Inspired by object detection algorithms in computer vision, we treat multi-tab WF recognition as a problem of predicting ordered sets with a maximum length. By adding enough single-tab queries to the detection model and letting each query extract WF features from different positions in the multi-tab traces, our model's Transformer architecture capitalizes more fully on trace features. Paired with our new proposed model training approach, we accomplish adaptive recognition of multi-tab traces with varying numbers of web pages. This approach successfully eliminates a strong and unrealistic assumption in the field of multi-tab WF attacks - that the number of tabs contained in a sample belongs to the attacker's prior knowledge. Experimental results in various scenarios demonstrate that the performance of TMWF is significantly better than existing multi-tab WF attack models. To evaluate model performance in more authentic scenarios, we present a dataset of multi-tab trace data collected from real open-world environments.

## CCS CONCEPTS

• **Security and privacy → Pseudonymity, anonymity and untraceability**; • **Networks → Network privacy and anonymity**.

## KEYWORDS

Tor; privacy; multi-tab website fingerprinting; Transformer

**ACM Reference Format:**
Zhaoxin Jin, Tianbo Lu, Shuang Luo, and Jiaze Shang. 2023. Transformer-based Model for Multi-tab Website Fingerprinting Attack. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3576915.3623107

## 1 INTRODUCTION

Tor [19] presently stands as the most popular low-latency anonymous communication network. It accomplishes the isolation of user IPs from server IPs by establishing encrypted tunnels across various network nodes, thus creating anonymity. Within the Tor network, user data traverses multiple intermediate nodes, referred to as "relays", for forwarding. Each relay only possesses knowledge of its preceding and succeeding nodes, but not the complete link information. Tor clients forward traffic through long-term encrypted tunnels called "circuits", with each circuit containing a series of three Tor relays: an entry relay, a middle relay, and an exit relay.

*Corresponding author: Tianbo Lu


*CCS '23, November26–30, 2023, Copenhagen, Denmark*

ACM ISBN 979-8-4007-0050-7/23/11…$15.00
https://doi.org/10.1145/3576915.3623107

While the aforementioned multiple layers of encryption and forwarding can, to some extent, prevent traffic data from being inspected, monitored, and tracked by third parties, adversaries who can observe communication between the Tor client and its entry relay can still launch targeted analysis attacks on encrypted traffic data, such as website fingerprinting (**WF**) attacks [1, 18]. In addition to being used by malicious attackers to identify the access targets of anonymous users, it can also assist law enforcement in tracking illegal activities on the dark web. By recording the sequence of traffic pattern information generated when visiting a website, such as packet sizes, directions, and timestamps, an attacker can conduct a macro-level analysis of website traffic patterns and obtain a summary of traffic patterns sufficient to uniquely identify the website's identity. Although WF attacks targeting Tor have achieved excellent performance in laboratory environments, their generalizability is challenged by various factors in real-world scenarios. [10, 31] have criticized some unrealistic assumptions made in research on WF attacks, among which the assumption of "single tab browsing behavior" can greatly impair the performance of WF recognition models in real scenarios.

In traditional WF attacks, researchers typically assume that users browse websites in a single tab, sequentially loading one webpage after another, and evaluate classification models using pure single-tab trace. However, this assumption is challenged by factors such as the slow webpage loading speed of the Tor browser [33] and the browsing habits of Tor users [5]. In fact, Tor users are likely to access the internet by opening multiple tabs in parallel.

In this paper, we propose a deep learning detection model called **TMWF** based on the Transformer [3] architecture for multi-tab browsing behavior. TMWF consists of two parts: DFNet [14] and Transformer. We chose DFNet as the feature extractor because it has achieved good performance in the single-tab website recognition task. As a typical CNN structure, DFNet effectively extracts spatial features from the original trace sequence by local modeling. Transformer is a general architecture for sequence prediction, and its self-attention mechanism can effectively model the global interactions between any elements in the sequence. TMWF uses DFNet to obtain the feature maps of the original trace sequence and inputs them as sequences to the Transformer Module. The module extracts the embedding of every single tab in parallel and maintains the positions of these embeddings relative to the specific tab in the original multi-tab trace. Inspired by object detection algorithms in computer vision such as DETR [12], we use $N$ single-tab queries as input to the Transformer decoder to extract single-tab features from the global features of the original traces. This operation enables the model to recognize the corresponding class of N sub-segments from any input sequence.

As an end-to-end multi-tab recognition model, TMWF does not rely on manual-designed features and only requires the use of raw multi-tab traces as input to identify each individual webpage trace from the mixed multi-tab traces. Specifically, the model consistently outputs $N$ prediction results, in which we classify all prediction results with unmonitored website classes and redundant predictions as "**no-tab**", and only retain prediction results with monitored website classes. Ideally, the $N$ correct website prediction results generated by the model consist only of the monitored class and redundant no-tab class. It's worth noting that the alteration in the output pattern, as described above, is facilitated by our new model training approach. The transformative contribution of the Transformer architecture lies in the enhancement of the performance of existing multi-tab WF attacks (such as BAPM [42]) through our proposed training approach.

Our main contributions are as follows:

- We have introduced a new training approach for multi-tab WF attacks, aiming to break the model's reliance on prior knowledge of the webpage number. For multi-tab WF samples with a randomly varying number of pages within a certain size range, this training approach generates ample prediction results and discards redundant "no-tab" classes. By doing so, the model achieves adaptive recognition of multiple monitored websites within traces.
- Our deep learning model, TMWF, is end-to-end and utilizes Transformer and learnable position encodings to automatically recognize single-tab WF from mixed multi-tab WF, effectively leveraging overlapping segments within the multi-tab WF. We use the complete multi-tab WF as input to the model, without requiring any manual adjustments on WF, until the final class prediction results are output.
- We propose a new set of evaluation metrics tailored to our proposed WF attack model. In the task scenarios that reflect the intentions of WF attackers, our new metrics can more accurately reflect the model's effectiveness. We compare TMWF with existing end-to-end multi-tab WF attacks on public datasets and our own datasets, and the evaluation results show that TMWF has better performance.
- We have collected a real open-world multi-tab trace dataset that includes non-monitored website traces. By randomly visiting over 6,900 non-monitored websites and 50 accessible monitored websites, we generated a multi-tab trace dataset under four different page-number settings within the range of 2 to 5.

The structure of this paper is as follows. In Section 2, we describe the threat model of WF attacks. In Section 3, we summarize related work and terminology on WF attacks and briefly introduce object detection techniques that inspired our design approach. In Section 4, we present the model architecture, and in Section 5, we describe the experimental setup and dataset construction. We analyze the experimental results in Section 6. In Section 7, we discuss the limitations of our work. In Section 8, we provide a summary of our work.

## 2 THREAT MODEL

In the Tor network, WF attacks typically occur on the link between the Tor client and the entry node [2], as shown in Figure 1.

The identity of the attacker includes but is not limited to the administrator of the LAN where the user resides, the administrator of the autonomous system or ISP, and the operator of the entrance node. Since traffic is encrypted, attackers cannot directly observe the target website that the user is accessing. However, they can identify the website's identity by analyzing the metadata of packets in the traffic: the attacker maintains a collection of monitored websites, accesses them in their controlled environment, and collects the resulting traffic traces. Based on these trace metadata, the attacker trains a WF classification model to match traffic traces with

website class labels. Subsequently, the attacker captures communication traffic generated by the targeted user when accessing a website and uses the classification model to identify whether the user has accessed websites within the monitored collection.

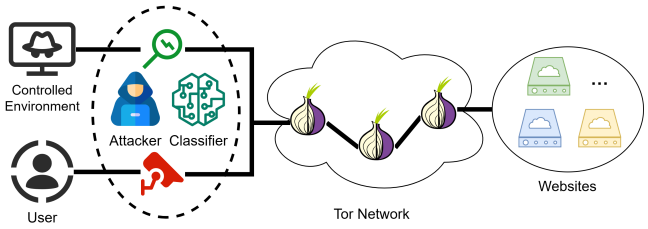



**Figure 1: Threat model of Tor WF attack.**

Research on WF models is typically evaluated in two scenarios: closed-world and open-world. In the closed-world evaluation, it is assumed that the client never accesses unmonitored websites, and the dataset used for the model only consists of traces from monitored websites. This evaluation is mainly employed for comparisons of scientific approaches in a controlled environment. In the open-world evaluation, unmonitored websites are introduced, and the client can access these websites. The dataset used for the model consists of traces from both the monitored and the unmonitored websites, with the number of unmonitored websites being much larger than the number of monitored websites, to simulate a broader range of websites that victims can access in the real world beyond the monitored websites.

Some research [2, 30, 36, 42] adopt a setup where the number of non-monitored websites is significantly greater than the number of monitored websites, but both have an equal total number of trace samples. This effectively assumes that the probability of a user accessing a monitored webpage is the same as accessing a non-monitored webpage. While this setup highlights classifier errors and facilitates clearer comparisons [30], it also carries the risk of introducing the base rate fallacy [10, 34]. This means that when the probability of access to the monitored websites is low, the attacker might be overwhelmed by false positive classifications produced by the model. Our experimental results in the paper also reflect this phenomenon.

In open-world evaluation scenarios, existing research often categorizes the model's task into binary and multi-class classification: both treat all samples from non-monitored websites as one class. However, in binary classification, all samples from monitored websites are also considered as a single class, while in multi-class classification, each domain's samples from the monitored set are treated as separate classes. Thus, multi-class models in open-world settings can be seen as variants of closed-world classification models, with an additional "non-monitored class" to encompass websites that attackers cannot monitor.

## 3 RELATED WORK

### 3.1 WF attacks

**Single-tab WF attacks.** Early WF attacks on Tor primarily employed machine learning models that relied on handcrafted features by domain experts, such as SVM [25], kNN [28], and CUMUL [2]. The work by [25] involved the extraction of Tor cells from TCP/IP packets, based on an understanding of the internal workings of Tor. For the first time, they processed packet sequences into directional sequences consisting of ±1 as input to the model.

Recent research in WF attacks has predominantly focused on deep learning methods. [30] demonstrated that deep learning models can implicitly extract features that are more resistant to concept drift. Building on this, [14] proposed DF, a CNN-based WF attack model that achieved excellent performance in both closed-world and open-world evaluations. In an effort to reintroduce some cumulative features into the model input, [21] developed Var-CNN based on a semi-automatic feature extraction process. [15] introduced TF, which applied n-shot learning to reduce the required number of training samples. Additionally, [20] showed that incorporating timing features can enhance the robustness of WF attacks.

To improve the precision of WF attacks at low base rates, [34] proposed three precision optimizers. [16] introduced the concept of a "Website Oracle" (WO), demonstrating that when combined with WF attacks, WO can significantly reduce false positives. Addressing the challenge of limited training samples, [13] introduced GAN-DaLF for performing WF identification. Meanwhile, [24] devised a robust traffic representation method coupled with a CNN-based classifier, achieving superior performance against state-of-the-art WF attack models. However, [4] concluded that simultaneously monitoring a large number of websites is likely impractical in real-world scenarios.

In the context of traffic fingerprinting, [17] proposed a hierarchical deep learning framework that bridges the gap between traffic heterogeneity and consistent neural network inputs. Furthermore, [7] revealed that different network congestion levels significantly affect the false-positive rate. In terms of WF security estimation, [29] presented a framework that generates tighter security estimates while reducing computational resources. [37] demonstrated that deploying padding-only defense methods across the entire network range also leads to increased latency. Finally, [11] critically assessed defense strategies against Tor traffic, specifically evaluating their effectiveness against the latest deep learning-based WF attack methods.

**Multi-tab WF attacks.** After [10] pointed out that multi-tab browsing behavior significantly degrades the performance of WF models under the single-tab assumption, [26] attempted to transform multi-tab traces into single-tab traces by splitting packet sequences and processing them using existing single-tab WF classifiers. They divided multi-tab traces into positive time intervals, zero time intervals, and negative time intervals according to the order of continuous access and attempted to split 2-tab traces with positive and zero time intervals using a time-based kNN algorithm. We adhere to the categorization of the organization of consecutive webpage traces resulting from multi-tab browsing behavior as outlined in [26]. As depicted in Figure 2, this categorization encompasses factors like the access interval, which denotes the waiting time between consecutive visits to two websites by the user.

[40] similarly used the splitting method to handle multi-tab traces. They proposed a BalanceCascade-XGBoost method to identify the boundary between the first and second pages from multi-tab traces, and then use an XGBoost-based website classifier to classify

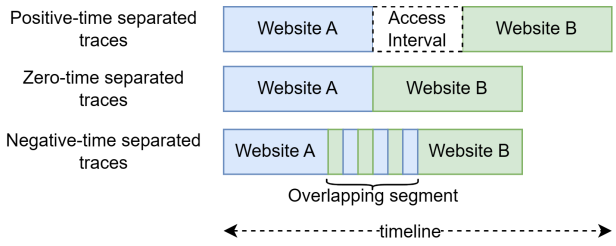



**Figure 2: Illustration of sequential traces' organizational patterns.**
**When a user accesses website B through a new tab before website A has fully loaded, it results in overlapping segments within the multi-tab traces generated by the browser.**

the pure segment of the segmented first-tab traces. Although their work further improved accuracy and expanded the experimental scenario to the open world, the objects that can be recognized are limited to only the first page in multi-tab traces.

[36] utilized a CNN model to differentiate between single-tab and multi-tab traces, and extracted a fixed number of packets from the beginning and end of 2-tab traces. They then used CNN, LSTM, and SDAE models to classify the extracted pure trace segments, further improving the accuracy of WF. However, their approach is still constrained by a limited number of page configurations and relies on multi-tab traces containing sufficiently long and non-overlapping pure segments.

BAPM [42] is the first end-to-end deep learning model that uses raw multi-tab trace as input and directly outputs the recognition results. It is also the first to utilize overlapping segments in multi-tab traces to assist the model in classification, achieving recognition of multi-tab traces with more than two pages while omitting manual operations. BAPM generates a tab-aware representation from directional sequences and performs block division, and then uses a self-attention mechanism to aggregate blocks with stronger relationships to facilitate website classification, which to some extent avoids the confusion effect caused by overlapping traces. [42] compared BAPM with three other multi-tab WF attacks and found that BAPM has the best and most stable performance.

Although significant progress has been made in multi-tab WF research, it has consistently relied on the "number of pages in input samples" as prior knowledge. Given that the number of pages contained in real-world multi-tab traces is unknown to attackers, models trained specifically for certain page numbers might suffer performance degradation when applied to traces with varying page numbers. To address this limitation, we propose a new training approach, coupled with our deep learning model, TMWF. Building on the achievements of BAPM's characteristics [42], we eliminate the reliance on "page number" as prior knowledge. Through this training approach, BAPM could achieve the same result as well.

Similar to [42], our research mainly focuses on multi-tab WF targeting overlapping webpage traces, as traces with positive and zero time intervals are easier to process [26]. We choose to utilize the overlapping segments within multi-tab traces in our deep learning model, rather than discarding this potentially valuable data as in some previous works. However, this also means we need to address the confusion effect caused by these overlapping segments on the classification model. In other words, we expect our research on WF deep learning models to extract useful information for distinguishing between two websites from these overlapping segments, rather than being confused by mixed information from multiple pages.

### 3.2 DETR

Object detection technology is an important technique in the field of computer vision. Its task is to detect objects of interest in images or videos and determine their location and category. Object detection technology is widely used in fields such as autonomous driving, security monitoring, and intelligent transportation [8, 12, 23, 41]. DETR [12] is a new and emerging object detection technology that uses a Transformer [3] architecture. It transforms the object detection task into a prediction problem for a set of objects. By using a set of learnable object queries to reason about the relationships between targets and global image context, DETR can directly predict the categories and positions of all targets in the input image without using anchor-based region extraction methods used in traditional object detection techniques. DETR simplifies the object detection process while maintaining good performance and generalization ability, and has a milestone significance in the development of object detection technology.

The inspiration behind our proposed TMWF model for multi-tab WF attacks comes from DETR. We input the feature sequence outputted by the DFNet framework into a Transformer encoder. By treating the monitored website's traces in the multi-tab traces as targets, we generate a sufficient number of tab queries, namely $N$ learnable position encodings (DETR sets $N$=100 by default, while we set $N$=6), for each sample. During training, the model continually updates the position encodings to learn information about the webpage positions from the features, ultimately enabling the model to recognize WF in multi-tab traces with various page settings (up to $N$ pages) in an adaptive manner.

## 4 MODEL ARCHITECTURE

### 4.1 Adaption for Transformer on multi-tab WF recognition

**Transformer.** Transformer [3] is a model for dealing with sequence problems, consisting of several encoders and decoders, where the encoder converts the input sequence into a series of context-aware representations, and the decoder uses these representations to generate the output sequence. Both the encoder and the decoder consist of the multi-head attention mechanism and the feed-forward neural network. The multi-head attention mechanism encodes the sequence, while the feed-forward neural network processes features at each position in the sequence. Additionally, Transformer introduces several other concepts and techniques, such as Positional Encoding. Positional Encoding embeds the position of each element in the sequence into a vector space, enabling the model to understand the positional information within the sequence. Benefiting from parallel computation capabilities, the capacity to handle long-range dependencies, and the ability to capture global context, Transformer exhibits remarkable scalability. It has found widespread application in fields like machine translation, text generation, and speech

recognition, solidifying its role as a fundamental architecture in the realm of natural language processing.

**Implementation method.** The realization of TMWF draws inspiration from the logical frameworks of DETR [12] and BAPM [42]. All three rely upon a foundational CNN architecture as the primary module of their model structures. This architecture is employed to extract latent features from the raw input sequences. These features are subsequently processed using other neural network structures to accomplish their respective tasks. Notably, DETR necessitates the recognition of object categories and coordinates for supervised learning. This involves configuring $N$ object queries, a number substantially larger than the object number in the ground truth samples. Each object query is designed to focus on specific feature positions. Consequently, DETR produces a fixed quantity $N$ of class prediction outcomes, coupled with their corresponding coordinate predictions. These predictions are then matched against the ground truth using a bipartite graph algorithm, thereby computing the loss. Due to the reliance on target coordinate annotations, the training approach of DETR proves challenging for the input sequences utilized by the WF model. In this context, the design concept of BAPM provided inspiration: by assigning fixed label identifiers to each attention head in the multi-head self-attention structure, the model only requires the sequence of page visits as supplementary supervision, enabling it to autonomously discern features among distinct pages.

Consequently, in TMWF, we incorporate learnable positional encodings (termed as "tab queries") to achieve a similar feature exploration capability. Specifically, we assign predefined label-associated feature regions for $N$ tab queries, unlike DETR's random allocation. Furthermore, to adapt TMWF for the multi-tab WF recognition task, we adopt DF's [14] major network structure as the local modeling backbone instead of ResNet [9] employed by DETR. To a certain extent, the overarching concepts of TMWF and BAPM align. Both models employ a fundamental CNN structure to extract latent features from the raw directional sequences. These features are then processed using other neural network structures, allowing them to generate a fixed number of predictions without relying on segment coordinate annotations within the samples. The performance enhancement of TMWF stems from the fine-grained feature utilization capacity of the Transformer.

## 4.2 TMWF

In this subsection, we will provide an overview of the architecture of TMWF. Additionally, for a more comprehensive understanding of the TMWF model architecture, further details are available in Appendix A. As depicted in Figure 3, TMWF consists of three main stages. The backbone module is responsible for extracting high-level feature representations from the raw input, which serves as the foundation for subsequent processes. The Transformer architecture is responsible for extracting WF embeddings for multiple pages from the contextual feature sequence, and the classification head then classifies multiple WF embeddings and outputs label predictions.

Although Transformer has shown strong performance in traditional AI applications, its use in the field of encrypted traffic analysis is still limited. In particular, CNN remains the main backbone for building models in WF research. This is because CNN performs well in capturing features between adjacent packets, which allows for effective modeling of local features. While CNN-based single-tab recognition models have shown strong performance in single-tab WF recognition tasks, their performance in multi-tab recognition tasks drops sharply. This is due to CNN's limited ability to capture long-range dependencies and global features, which makes it difficult to distinguish between features of different pages in more challenging multi-tab website recognition tasks.

The task of multi-tab WF recognition involves identifying the website corresponding to each webpage from the traces formed by the traffic of multiple pages. The ground truth of each multi-tab trace can be viewed as a set of labels, where the $i$-th element represents the website to which the $i$-th page trace belongs. Therefore, the multi-tab website recognition task can be treated as a set prediction problem.

**Backbone.** In TMWF, we utilize DFNet to perform local modeling on the input multi-tab traces. DFNet transforms the input multi-tab traces $\mathbf{wf}$ into a sequence of WF features $F$. We denote the major network structure of DF [14], excluding the classification head, as DFNet. Here, $L$ and $l$ respectively represent the sequence length before and after being processed by DFNet.

$$F = DFNet(\mathbf{wf}), \mathbf{wf} \in R^L, F \in R^{l \times d} \tag{1}$$

**Transformer Encoders.** We use vanilla Transformer encoders to model the normalized feature sequence $F^{'}$ to extract the WF context feature sequence $Z$. Although the features $F$ output by the backbone carries positional relationships, the Transformer lacks inductive bias towards positional information in its design and cannot learn positional information from input features. To compensate for this deficiency, we add learnable positional encoding $P$ to the feature sequence outputted by the backbone and use the sum as the input to the Transformer encoders. Assuming that the number of encoders is $N_E$, the operation process is as follows:

$$\begin{gathered} F^{'} = LayerNorm(FW + \mathbf{b}) + P, \\ F^{'} \in R^{l \times d}, P \in R^{l \times d}, W \in R^{d \times d}, \mathbf{b} \in R^d \end{gathered} \tag{2}$$

$$\begin{gathered} O_i = TransformerEncoder(O_{i-1}), \\ O_0 = F^{'}, Z = O_{N_E}, i = 1 \cdots N_E \end{gathered} \tag{3}$$

**Tab Queries.** Although the Transformer Encoders are effective in global modeling, their output context feature sequence do not separate the features of different pages in multi-tab traces. To this end, we introduce $N$ tab queries (corresponding to object queries in DETR) as references for the Transformer decoders to query the fingerprint features of different pages. We use $T_Q \in R^{N \times d}$ to denote the tab queries.

**Transformer Decoders.** We use vanilla Transformer decoders to extract the WF features of $N$ pages from the contextual feature sequence $Z$ output by the encoders. The decoders can find a plausible alignment between tab queries and contextual feature sequence so that each tab query finds the fingerprinting feature of the page to which it belongs. We use $E_{WF}$ to denote the fingerprint features of $N$ pages. Assuming that the number of decoders is $N_D$, the operation process is as follows:

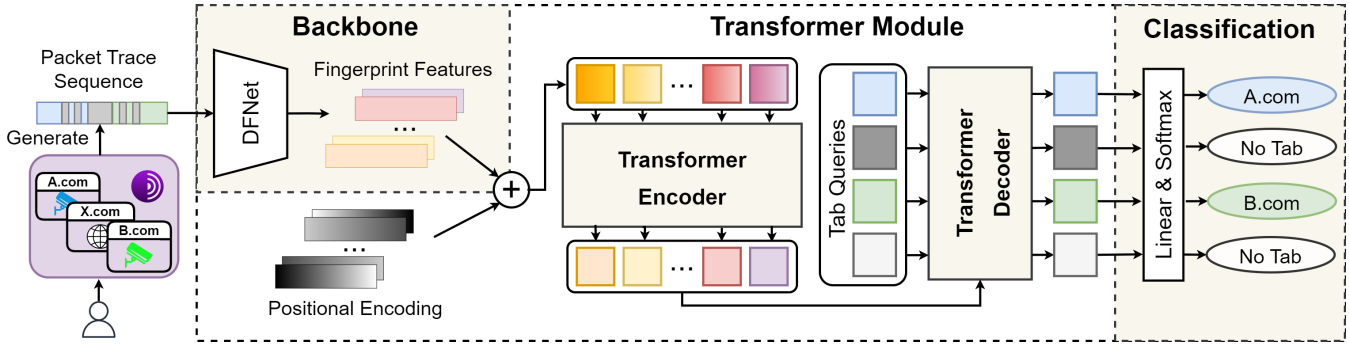



**Figure 3: Workflow of TMWF configured with $N = 4$.**

$$O_i = TransformerDecoder(O_{i-1}, T_Q),$$
$$O_0 = Z, E_{WF} = O_{N_D}, i = 1 \cdots N_D \tag{4}$$

**Classification.** We use a classification head to classify the WF embeddings of $N$ pages extracted by the Transformer decoders. This classification converts each page embedding into a probability distribution over websites, and the website with the highest probability is considered the predicted result of TMWF for that page. Assuming there are $C$ websites in total, the above process can be described as:

$$Pr = Softmax(E_{WF}W_5 + \mathbf{b_5}), W_5 \in R^{d \times C}, \mathbf{b_5} \in R^C, Pr \in R^{N \times C} \tag{5}$$

The $i$-th row of matrix $Pr$ represents the probability distribution over websites for the $i$-th page. While TMWF produces $N$ website prediction results, we expect that the number of pages in the actual prediction results of the model is typically no more than $N$. For non-monitored websites and redundant prediction results, we uniformly assign the label "**no-tab**".

# 5 EXPERIMENT

## 5.1 New training approach

We draw an analogy between monitored websites in WF attacks and recognizable objects in object detection. For multi-tab trace samples with varying numbers of pages, we assign $N$ labels to each sample. When the original number of labels is less than $N$, we augment the sample's label number with "no-tab" class labels. Following this design philosophy, these annotations guide the model during training to learn features conducive to distinguishing between different pages. As depicted in Figure 3, when using the $N = 4$ setting, for a multi-tab trace sample containing traces from two monitored websites, ideally, the first and third predictions among the four generated by the model will capture segments belonging to the monitored websites within the sample. For test set samples, the number of targets it contains is transparent to the model, and the model will always output $N = 4$ predictions. The model's owner needs only select predictions with predicted classes other than "no-tab" as the final results. This approach thus achieves adaptive recognition for a range of target quantities (from 0 to 4). In essence, the removal of prior knowledge regarding the actual target number in input samples comes at the cost of generating redundant results beyond the actual number of labels.

In theory, the training approach we propose can be applied to multi-tab WF attack models like TMWF and BAPM, which have a fixed number of output predictions. This enables them to achieve adaptive recognition of monitored websites within multi-tab traces with an upper limit of $N$ pages. In our research, we practically set $N = 6$. The reason for choosing this number is due to the limitation of our hardware conditions, which restricts us from synthesizing multi-tab WF datasets with more than 6 pages. In real-world scenarios, this number can be set higher to meet the needs of the attacker.

Strictly speaking, models still rely on knowing the maximum number of tabs. In situations where the attacker sets the maximum tab number $N$ lower than the actual tabs visited by a user, the model may not recognize all the websites within a complete trace. In such cases, the $N$ predictions of model outputs would ideally correspond to the first $N$ sub-segments of the trace, each associated with a website. However, we view this as a soft constraint that can be alleviated by setting a sufficiently large value for $N$ and employing techniques like sliding windows for sampling. This ensures that the model can accommodate various scenarios effectively.

## 5.2 New evaluation metrics

Previous multi-tab WF research [36, 42] relied on the condition that the number of labels (tabs) in the ground truth matches the number of predicted results. This approach required that all samples in the test set have a fixed number of labels enabling the calculation of scores for each page individually. However, this evaluation method cannot be extended to multi-tab trace test sets containing varying numbers of pages.

Moreover, existing metrics treat all non-monitored traces as a single class, including them in the calculation process. However, considering the attacker's actual intent to identify monitored websites, and to more accurately evaluate the model's performance in this task, we follow the evaluation approach used in object detection domains[1]. We do not incorporate correctly predicted results for non-monitored set pages into the metric calculation process.

This means that for a multi-tab trace consisting of $s$ webpages containing $t$ unmonitored webpages ($s \geq t$), we only consider the

[1]https://cocodataset.org/#detection-eval

positive predictions (predicted class belongs to the monitored set) and false negative predictions (predicted class belongs to the monitored set, but the ground truth belongs to non-monitored set) from all model predictions as "valid predictions". These valid predictions are compared against the ground truth of the samples. Otherwise, due to TMWF's strong ability to recognize these filled "no-tab" labels (as shown in Figure 6), our model's evaluation scores would be inflated by these redundant results. We define "redundant results" as correct predictions among the part of multiple predictions output by the classifier that exceeds the actual tab number in the original sample (i.e., manually padded labels). These predictions have a label "no-tab". This definition is only used in the calculation process of our metrics, and the classifier does not need to be aware of whether its predicted results are redundant. We refer to these metrics as **Overall Accuracy**, **Overall Precision**, and **Overall Recall**.

We have categorized our newly proposed metrics into two distinct configurations: lenient, denoted as **"Basic"**, and stringent, referred to as **"Advanced"**. In both of these configurations, we assume that the number of website pages contained within the captured traffic traces, from an attacker's perspective, remains a priori unknown. We introduced the "Advanced" metrics for a specific reason: essentially, our model's output predictions possess a sequential order, yet the "Basic" metrics overlook this attribute during computation. While [42] and [36] evaluated the independent performance of the model on each page of the multi-tab traces, even though it wasn't explicitly covered in our experiments, we aimed for the "Advanced" metrics to reflect the model's capability to recognize the sequence of sample pages within the predicted results.

**Basic metrics.** We assume that the attacker is only interested in identifying whether the victim has visited any websites in the monitored set, without considering the order of the identified websites. Both the ground truth and the positive prediction results are transformed into sets of unique labels when calculating the metrics. The relevant formulas for the Basic setting are as follows:

$$Accuracy_{Basic} = \frac{\sum_{i=1}^{n} |T_i \cap P_i - \{\text{"no-tab"}\}|}{\sum_{i=1}^{n} max(|T_i - \{\text{"no-tab"}\}|, |P_i - \{\text{"no-tab"}\}|)} \quad (6)$$

where $n$ is the number of samples in the dataset, $T_i$ and $P_i$ are the ground truth set and prediction set for the $i$-th sample, respectively, and "$|\cdot|$" is used to calculate the number of elements in a set.

$$Precision_{Basic} = \frac{1}{m} \sum_{j=1}^{m} \frac{TP_j}{TP_j + FP_j} \quad (7)$$

$$Recall_{Basic} = \frac{1}{m} \sum_{j=1}^{m} \frac{TP_j}{TP_j + FN_j} \quad (8)$$

Where $m$ is the total number of classes excluding the "no-tab" class, $TP_j$ and $FP_j$ represent the number of true positives and false positives in class $j$, and $FN_j$ is the number of false negatives in class $j$.

**Advanced metrics.** Assuming the attacker's objective is not only to identify the types of websites visited by the victim but also their order of visitation and frequency, the ground truth and positive prediction results will be considered as equally long label lists that can contain duplicate elements (the "no-tab" label will be used to fill in the list if necessary). The following formulas are based on the Advanced setting:

$$Accuracy_{Advanced} = \frac{\sum_{i=1}^{n} \sum_{k=1}^{S} [p_{ik} = t_{ik}, t_{ik} \neq \text{"no-tab"}]}{\sum_{i=1}^{n} (S - \sum_{k=1}^{S} [p_{ik} = t_{ik} = \text{"no-tab"}])} \quad (9)$$

Where $n$ represents the number of samples in the dataset, $S$ represents the length of the label list after filling, and $p_{ik}$ and $t_{ik}$ represent predicted labels and true labels, respectively. The symbol "$[\cdot]$" is used to count the number of elements in a list that meet certain conditions. When the expression inside the brackets is true, the result of "$[\cdot]$" is 1, otherwise, it is 0.

$$Precision_{Advanced} = \frac{1}{m} \sum_{j=1}^{m} \frac{tp_j}{tp_j + fp_j} \quad (10)$$

$$Recall_{Advanced} = \frac{1}{m} \sum_{j=1}^{m} \frac{tp_j}{tp_j + fn_j} \quad (11)$$

Where $m$ represents the total number of classes excluding the "no-tab" class. $tp_j$ and $fp_j$ respectively represent the true positive and false positive counts for class $j$, while $fn_j$ represents the false negative count for class $j$.

To aid in illustrating the intuition behind our proposed new metrics, we present a simplified hypothetical example of the evaluation process in Appendix B.

## 5.3 Experiment design

We have selected BAPM as the baseline for evaluation because it is the first end-to-end model for multi-tab WF and outperforms previous works in experimental performance. We then made modifications to the BAPM model to align it with our proposed new training approach. The evaluation was conducted within a more realistic task scenario where the number of pages composing the input sequence for the model is unknown to the attacker. We refer to our modified version as Adaptive-BAPM. Therefore, both Adaptive-BAPM and TMWF possess the capability to recognize multi-tab traces with any number of pages. Adaptive-BAPM achieves this using $N$ attention heads, while TMWF employs $N$ tab queries.

In the original BAPM, the number of attention heads used by the attacker when creating the model is the same as the number of pages in the input sample. This means that regardless of the number of pages in the input sample, the number of predicted results generated by the model is always equal to the predetermined number of attention heads. As the range of fluctuations in the number of web pages in the input sample increases, the performance of the model further deteriorates. In Adaptive-BAPM, we adjust the number of attention heads used by the model to six in an attempt to improve this flaw. This number exceeds the number of pages contained in any dataset sample we used in the experiment. This means that regardless of the number of pages in the input sample, the model generates six predicted results. Ideally, the portion of labels in the predicted results that exceed the number of pages contained in the

sample should be the "no-tab" class. To investigate the performance of our proposed TMWF, we designed the following experiments:

**(1) Model design, validation, and exploration of synthetic traces.** We deviated from the typical model design and validation approach commonly employed in previous WF studies (parameter tuning on closed-world datasets). Leveraging our new training approach, we directly adjusted certain parameter settings using manually synthesized open-world datasets. Same with BAPM[42], we used the Walkie-Talkie [27] dataset to manually synthesize multi-tab traces. Consistently, we trained the model on a synthesized 6-tab dataset and fine-tuned TMWF parameters on a corresponding 6-tab test set. In addition, we attempted to introduce timing information as model input and conducted ablation experiments to demonstrate the rationality of our model architecture.

To illustrate the adaptive recognition capacity of both Adaptive-BAPM and TMWF for samples of diverse page numbers, tests were conducted using 2-tab, 4-tab, and 6-tab traces. These evaluations were performed using both conventional metrics and the new metrics proposed by us, allowing for a comprehensive comparison of model performance. Given the temporal gap in data collection for the Walkie-Talkie dataset, we replicated a subset of experiments on our self-collected dataset. Detailed experimental results can be found in Appendix F.

Due to the potential disparities between traces gathered from real-world environments and manually synthesized traces, we conducted training experiments using both real 2-tab traces and 2-tab traces generated through various synthesis methods. Simultaneously, we altered the size of the synthetic training set. We maintained consistency by evaluating all models on real 2-tab traces to assess the impact of these different traces and training set sizes on model performance. The corresponding experimental process can be seen in the Appendix E.

**(2) Closed-world Experiments.** To demonstrate that our modified Adaptive-BAPM did not exhibit a significant performance decrease, we compared TMWF, the original BAPM, and Adaptive-BAPM on the real-world dataset published by [42].

**(3) Open-world Experiments.** We collected real multi-tab traffic traces generated by the Chrome browser (similar experiments were conducted using traces from the Tor browser as well, and the corresponding results are provided in Appendix H). The set of randomly accessed websites consists of 50 monitored websites and over 6900 non-monitored websites, with page numbers of 2, 3, 4, and 5.

Based on the findings from the impact of multi-tab trace synthesis methods on model performance in **(1)**, we utilized the synthesis method that yielded the most significant improvement in model performance for creating the synthetic multi-tab trace training set. Additionally, we employed various synthesis methods to create synthetic multi-tab trace validation sets. By comparing model performance on multiple synthetic validation sets and real multi-tab trace test sets, we evaluated the differences in performance.

We also attempted an analysis of the model's prediction results. Moreover, guided by the results of these experiments, we selected a synthesis method that best approximated the results of real multi-tab test sets for creating simulated test sets with varying ratios of monitored and non-monitored traces (considering different base rates for monitored websites). Subsequently, we evaluated the model's performance under these conditions in Appendix I.

**Table 1: Parameter settings for TMWF.**

| Model Part | Details | Value |
|---|---|---|
| DFNet-CNN | Input Dimension | (30720,1) |
| | Kernel Number | [32,64,128,256] |
| | kernel Size | [8,8,8,8] |
| | Pool Size | [8,8,8,8] |
| Transformer | Input Dimension | (121,256) |
| | Encoder Number | 2 |
| | Decoder Number | 2 |
| | Head Number | 8 |
| | Feed-forward Dimension | 1024 |
| | Tab Query Number | 6 |
| | Dropout Rate | 0.1 |

For all of our experiments, we maintained the parameter settings used by TMWF as shown in Table 1 (obtained through the parameter tuning process described in Appendix C). For the Adaptive-BAPM used in comparisons, we only modified the number of attention heads, the length of a single page, and the block length, while keeping the other parameters the same as in the original paper.

## 5.4 Dataset

In the existing research on multi-tab WF attacks [36, 42], models are trained using manually synthesized multi-tab traces from raw single-tab traces, which we have also followed. This means that by selecting different domain names from the single-tab trace dataset and setting the overlap ratio of the traces, an attacker can obtain simulated multi-tab traces under various conditions. Although intuitively, training the model using traffic data generated from real multi-tab browsing behavior allows the model to learn more implicit representations of real traffic trace features, this comes at a cost of exponentially increasing expenses in collecting multi-tab traces relative to the manual synthesis of multi-tab traces using a small number of single-tab trace samples. Furthermore, as the "length of the trace of each tab" is posterior in collecting traffic, it is difficult for an attacker to precisely control the overlap ratio[2] between the collected multi-tab traces [42]. A detailed description of the manual synthesis method can be found in Appendix D.

We refer to the method of synthesizing multi-tab traces based on the overlap ratio as "$M_{ratio}$". Furthermore, we have devised a new synthesis method that aligns with the access patterns used during the collection of real multi-tab traces. This involves introducing varying time delays to multiple single-tab traces (where the added delays accumulate) and then concatenating them to form a complete multi-tab trace. This new synthesis method is termed "$M_{delay}$". It's important to note that $M_{delay}$ is primarily designed to simulate the access mode of multi-tab traces as observed in real-world collections. However, it introduces additional uncertainty, such as instances where traces do not overlap or when traces from one page are fully obscured by traces from another page. Therefore, for the purpose of validating and comparing model performance, we continue to use

[2]We define the overlap ratio as the proportion of the length of segments involved in the mixed part with the second page to the length of the first page.

$M_{ratio}$ for synthesizing multi-tab traces. This is done to maintain a consistent working environment as in the existing work [42]. Subsequently, we apply our newly designed $M_{delay}$ to synthesize multi-tab traces for other scenarios to assess TMWF's performance. For a more comprehensive exploration of the impact of synthesized traces versus real traces on model performance, please refer to the details provided in Appendix E.

Although manually synthesized multi-tab traces are convenient for training models, the distribution of features exhibited by real multi-tab traces may differ, resulting in a degradation of model performance. Some research [36, 42] have noticed this issue and collected additional traffic generated by real multi-tab browsing behavior to evaluate model performance in more realistic scenarios. In our experiments, in addition to using the real multi-tab trace dataset published by [42] to test the performance of our proposed models, we also constructed multiple real open-world datasets with different page numbers and containing non-monitored webpages to investigate the generalization performance of our proposed model, as well as the performance differences between the simulated validation set and the real-world test set.

$D_{WT}$**.** To compare the performance of our proposed model with existing multi-tab WF attack models, we used the single-tab trace dataset of Walkie-Talkie [27] to manually synthesize multi-tab trace datasets for the training and evaluation processes of our model, same as [42]. To achieve this, we employed a similar strategy to theirs by selecting 50 websites from the original dataset with the most data packet traces. We used 80% (about 90 traces) of each website's samples in the monitored set to synthesize the multi-tab trace training set, while the remaining samples were used to synthesize the testing set. It's crucial to note our initial partitioning of the original single-tab trace dataset into two subsets, maintaining a 4:1 total ratio proportion while preserving a consistent balance of sample numbers across classes. This two-step process enables us to synthesize both the training and testing sets of multi-tab traces based on the respective subsets of single-tab traces. This approach ensures the isolation of website samples between the training and testing sets, wherein only the domains of monitored websites are recurrent across the training and testing sets.

We employed the $M_{ratio}$ approach to generate overlapping traces, adopting a more randomized methodology for selecting constituent samples in the synthesis process. Specifically, we amalgamated all monitored website samples and non-monitored website samples into a candidate trace list. Subsequently, we employed a completely random selection process from this list to pick multiple single-tab traces. We randomly sampled values within the range of [0.1, 0.2, 0.3, 0.4, 0.5] as the ratio for overlapping regions. To prevent the complete overlay of traces, we set a purity segment threshold of 0.1 for intermediate webpages with webpages both before and after. Additionally, we removed the constraint that "constructed traces must include monitored websites", implying that the synthesized multi-tab traces could include consecutive monitored websites, consecutive non-monitored websites, or both.

We believe that this strategy generates datasets more closely aligned with network traffic patterns an attacker could feasibly monitor. This approach also allows us to increase the scale of the synthetic dataset to five times the size of the original single-tab dataset, resulting in approximately 22,000 samples in the training set and 5,500 samples in the testing set. The trace information within the dataset encompasses not only the direction sequence of data packets but also the timing information sequence.

In the subsequent sections, we employ the notation **"page number-dataset usage"** to denote the current dataset. For instance, we use notations like $D_{WT-6tabs-train}$ and $D_{WT-6tabs-test}$ to refer to the training and testing sets of 6-tab traces, respectively, synthesized from the single-tab trace dataset provided by [27] using the $M_{ratio}$ approach.

$D_{BAPM}$**.** We elected to conduct closed-world experiments on the real-world dataset presented by [42], in order to substantiate that our modified Adaptive-BAPM does not exhibit a significant performance degradation. Within this dataset, we performed comparisons among TMWF, the original version of BAPM, and the Adaptive-BAPM. Both the training and testing sets of this dataset were derived from 50 monitored websites. The training set comprises 10,000 traces, with each website contributing 200 traces. The testing set consists of 1,000 2-tab traces, all of which are real multi-tab traces. The traces are presented in the form of Tor cell direction sequences composed of ±1 values, where 0 signifies padding.

$D_{CHR}$**.** In reference to the data collection methodology outlined in [30, 36, 40, 42], we adopted the following scheme for data acquisition: On three VPS hosts located in the United States, we created a Docker image integrating the Selenium browser automation tool, the Chrome browser, and the Tor proxy service. We used this image to create several docker containers. In each docker container, we started an independent Tor proxy service and used a Selenium script to drive the Chrome browser to automatically visit websites in headless mode. We set the maximum loading time for all websites to 30 seconds. After each visit, we closed the browser and cleared the cookies, and waited for 2 seconds before starting the next capture.

We retrieved the website ranking list[3] [32] on July 18th, and manually selected 50 accessible websites from the top 100 ranked sites for our monitored set. The following 9900 sites were designated as the unmonitored set. Within the monitored set, each website was visited 200 times, while within the unmonitored set, each website was visited only once. Utilizing network namespaces, we captured the traffic generated by the Tor proxy service within each Docker container using tcpdump. Subsequently, employing Wireshark, we extracted TLS packets exceeding 512 bytes. Following the methodology outlined in [25], we extracted the Tor cell direction sequence. Additionally, we also saved the time sequence for generating multi-tab traces.

After collecting traces generated by single-tab visits through the aforementioned method, we identified over 6,900 accessible websites. These websites were utilized as the filtered unmonitored set. We conducted data collection with varying sequences of 2, 3, 4, and 5 consecutive page visits, resulting in the collection of approximately 1000 multi-tab traces for each sequence length. Given the challenges associated with controlling the proportion of overlap between multi-tab traces during real-world data collection, a more practical approach was adopted: introducing a predefined time delay between the sequential visits to two different web pages. For a given dataset of multi-tab traces with $N$ pages, the script

[3] https://trancolist.eu/list/

generates a genuine multi-tab trace as follows: (1) Randomly select $N$ websites from the list of all accessible websites. Each monitored website in the list contains multiple duplicate elements, while each unmonitored website contains only one duplicate element so that monitored and unmonitored websites are equally likely to be selected during random selection. (2) Open the browser and visit the first website, followed by a random wait time[4] between 2 to 6 seconds. Subsequently, a new tab is opened in a separate thread to access the second website. Another random wait time between 2 to 6 seconds is introduced. If $N = 2$, the browser is closed and cookies are cleared after the second page loads. If $N$>2, an additional wait time of between 2 to 6 seconds follows, then a new tab is opened to access the third website. This process is repeated iteratively for $N$ consecutive page visits.

Following the process, the generated multi-tab traces ideally include a pure segment on each page that has not been involved in overlapping with other pages (although exceptions to this scenario can't be completely ruled out, as mentioned earlier). This is visually depicted in Figure 4. We assume that users typically do not rapidly open multiple tabs in a way that leaves any page within the multi-tab trace without a pure segment.

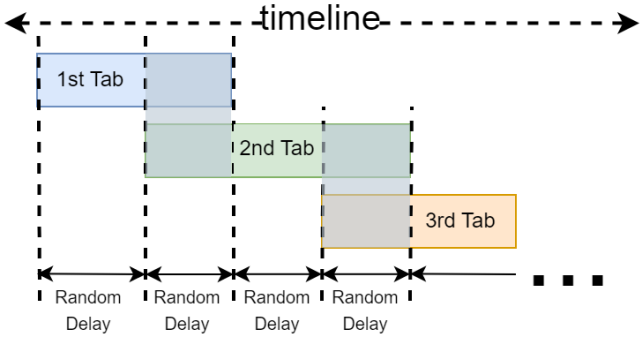



**Figure 4: An ideal illustration of a real multi-tab trace.**

We conducted the collection process between July 22, 2023, and August 5, 2023. The collected single-tab traces were used to synthesize multi-tab traces. For monitored websites, we selected 100 traces from each site, with 80 for training and 20 for validation. For non-monitored websites, only traces with more than 50 packets were retained, as shorter traces were insufficient to represent overlapping phenomena.

Consistent with the $D_{WT}$ approach, we divided traces of all classes into two subsets in a 4:1 ratio. Subsequently, we synthesized training and validation sets based on these subsets to maintain sample separation between the two. As various multi-tab trace synthesis methods may be involved in different task scenarios, we used a naming convention like **"page number-synthesis method-dataset usage-sample number"** to differentiate the synthesized multi-tab trace datasets[5]. For example, $D_{CHR-6tabs-ratio-train-40k}$ represents a training set of 6-tab traces synthesized using the $M_{ratio}$ method from the 4/5 subset of the original Chrome single-tab trace dataset (size 8000, comprising 50 monitored websites with 80 traces each and 4000 non-monitored website traces), with a total sample count of 40000. For real multi-tab trace test sets, we consistently use the notation **"page number-real-sample number"** to represent these datasets.

Furthermore, based on the finding in Section 6.4 that the model performance using multi-tab traces synthesized with random delay as a validation set may be similar to the real multi-tab trace test set, we attempted to explore TMWF's recognition capability for monitored websites with different base rates using the $M_{delay}$ synthesized multi-tab trace test set in Appendix I. Following [34], we defined $r$ as the ratio between the frequency of client visits to non-monitored pages and the frequency of client visits to monitored pages (equivalent to visiting a monitored page once every $r$ visits to non-monitored pages). Higher $r$ values would lower precision and make the open-world classification problem more challenging. We controlled the ratio of non-monitored to monitored traces to be $r$ and evaluated the model's performance for three cases: $r = 1$, $r = 10$, and $r = 100$.

We employed the newly designed $M_{delay}$ method to synthesize multi-tab traces. During random sampling, the selection probability of non-monitored traces was set to $r$ times that of monitored traces. When $r = 1$, we synthesized 500 multi-tab traces for each page number from 2 to 5; when $r = 10$ and $r = 100$, we synthesized 5,000 and 50,000 multi-tab traces for each page number respectively. For each $r$ value setting, all the synthesized multi-tab traces were combined into a new multi-tab trace dataset, referred to as $D_{CHR-mixed-delay-r1}$, $D_{CHR-mixed-delay-r10}$, and $D_{CHR-mixed-delay-r100}$.

$D_{TBB}$. $D_{TBB}$ dataset was collected following the same approach as $D_{CHR}$, but using Tor Browser Selenium[6] to capture traffic generated by the Tor Browser Bundle (TBB) while browsing websites. We chose to collect this dataset because the Tor browser implements a specific policy - it isolates traffic based on the first-party domain in the URL bar. This means that if a user loads two tabs with different domains, the traffic generated by these two pages will be routed through different circuits[7]. To explore the impact of differences in traffic between the Chrome and TBB environments on the model, we conducted experiments similar to those described in Section 6.1 and Section 6.4, each detailed in Appendix F and Appendix H.

# 6 ANALYSIS

## 6.1 Model performance validation

We used the $D_{WT-6tabs-train}$ dataset, which contains approximately 22,000 samples, to train all models. Then, we evaluated the performance of these models on the $D_{WT-2tabs-test}$, $D_{WT-4tabs-test}$, and $D_{WT-6tabs-test}$ datasets, each containing around 5,500 samples. The evaluation results for the same experiments conducted on the $D_{TBB}$ dataset are presented in Appendix F. We evaluated model performance using both the metrics from the existing work [36, 42] and our proposed new metrics. For input samples with less than 6 tabs, we uniformly padded them with a "no-tab" label.

[4]This time interval is estimated based on the quartiles of the website loading times recorded during the collection of single webpage traces.

[5]The synthesized multi-tab trace datasets notated as "val" and "test" in this paper both utilize a 1/5 subset of the original single-tab trace dataset as their data source.

[6]http://github.com/webfp/tor-browser-selenium

[7]https://2019.www.torproject.org/projects/torbrowser/design/#identifier-linkability, https://gitlab.torproject.org/tpo/applications/tor-browser/-/issues/18311

The evaluation results for [36, 42] metrics based on accuracy, precision, and recall are referred to as Previous, while our proposed new metrics are named Overall Basic and Overall Advanced. Previous metric results are a series of numbers, where each number represents the model's score on the corresponding page. Since our models always output 6 prediction results, we manually removed the redundant scores for the 2-tab and 4-tab test sets. The specific results are presented in Table 2.

(1) The evaluation results obtained from Adaptive-BAPM using the old metrics show that the model can recognize the first webpage of multi-tab traces with relatively high accuracy. However, in the evaluation results of the subsequent pages, although accuracy is maintained between 45% and 50%, precision and recall have dropped to an unusable level. We speculate that the reason for this phenomenon is that Adaptive-BAPM still has some ability to distinguish monitored websites from unmonitored websites, but it is powerless to further subdivide the classes within monitored websites. The relatively stable fluctuation range of accuracy is contributed by the correct predictions of the model for unmonitored websites. However, calculating the precision and recall for each page, involves taking the average of the results for each class, and since the evaluation results of the monitored website class, which accounts for the absolute majority, are poor, the overall results are maintained at a low level. Since we used a synthesized 6-tab trace dataset for model training, when evaluating Adaptive-BAPM using the old metrics on the 6-tab test set, it is essentially equivalent to using BAPM for the 6-tab experiment. However, compared to the results presented in the original paper [42] for the 3-tab experiment, our results have significantly degraded performance. After comparing the evaluation results of the original paper's 2-tab experiment and 3-tab experiment, we believe that this performance gap is not only caused by the increase in the number of pages. We hypothesize that the primary reason behind this phenomenon lies in the substantial confusion effect introduced by the open-world non-monitored websites integrated into our multi-tab dataset. Furthermore, compared to the original work using 10% as the overlap proportion between any adjacent page in the 3-tab experiment, we use random values in [10%, 20%, 30%, 40%, 50%] as the overlap proportion. This setting leads to traces in the middle of the multi-tab trace having shorter pure segments, and pure segments are an important basis for BAPM classification, making it difficult to maintain its original high performance. In contrast, TMWF shows higher robustness in dealing with overlapping traces: the numerical levels of each metric on pages 2 to 5 are not significantly different from those on the first and last pages.

(2) As non-monitored webpages constitute half of the total number of webpages in the multi-tab trace, the prediction results of non-monitored traces have a significant impact on the final evaluation results (especially accuracy). Our proposed new metric more fully reflects the needs of real-world attackers to identify the monitored websites, compared to the traditional metrics that incorporate the correct prediction of non-monitored webpages in the calculation process.

(3) TMWF achieved the highest accuracy on all metrics. Furthermore, the evaluation results on the 2-tab and 4-tab test sets demonstrate TMWF's excellent generalization ability. TMWF can adaptively identify samples with different numbers of web pages. We observed that for both Adaptive-BAPM and TMWF when using a test set with a different number of pages than the number of pages in the training set, the larger the page number, the greater the decline in the performance of each page represented by the old metrics. This phenomenon is particularly evident in the evaluation results on the 4-tab traces. However, TMWF achieves good performance on the **Overall Basic** metric, possibly due to the redundant predictions generated by the model containing classes that are identical to those in the ground truth list, leading to only a slight decrease in TMWF's overall score. Overall, if we focus only on identifying the types of websites that users access, without considering whether the model's output predictions exactly correspond to the user's browsing process, TMWF's high generalization ability and performance on the Overall basic metric demonstrate its ability to successfully perform this task.

(4) Upon careful scrutiny of the results obtained from the Previous metric, it is discernible that, for both Adaptive-BAPM and TMWF, the classifier's attack performance for tab $N$ ($N$>1) experiences a decline with the increase in page order (excluding the final page). While intuitively one might anticipate that these page trace segments could possess similar overlapping patterns, and such similarity should not lead to a degradation in model performance, we posit that the fundamental reason for this phenomenon lies in the disparate lengths of traces (directional sequences) from distinct websites. The variance in these lengths engenders a greater dispersion in the positioning distribution of later-traversed traces in the synthesized sequence during multi-tab trace synthesis. Consequently, the challenge is heightened in effectively learning the "shared spatial information representation of the current page from all samples through the corresponding tab query for the $i$-th page" (for Adaptive-BAPM, this pertains to inter-relationships among distinct feature blocks). This intricate difficulty ultimately culminates in the decline of attack performance.

## 6.2 Ablation experiment

We consider the Transformer architecture to be the most critical component of TMWF, as it effectively captures dependencies between different positions by performing self-attention calculations on the input features. In addition to the Transformer architecture, we also conducted ablation analysis experiments on the feature extractor and input data in TMWF. Same with Section 6.1, we maintain the uniform use of the $D_{WT-6tabs-train}$ dataset for model training. The experimental results of the $D_{WT-6tabs-test}$ are presented in Table 3. The "Add Timeinfo" experiment indicates our exploration of incorporating timing information sequences as inputs to the model. The "Original model" refers to using only the packet direction sequences as inputs to the complete model. "No DF" indicates that we replaced TMWF's feature extractor with the CNN-based feature extractor used in BAPM. "No Transformer" indicates that we removed the Transformer architecture from TMWF and imitated the operation of [42] by using sigmoid instead of softmax activation function in DF and using binary entropy loss function instead of the categorical entropy loss function, directly predicting the input samples based on the output feature map of the feature extractor, and obtaining multiple website labels based on the maximum probability value.

**Table 2: Evaluation of prediction on $D_{WT-2tabs-test}$, $D_{WT-4tabs-test}$, and $D_{WT-6tabs-test}$ based on models trained with $D_{WT-6tabs-train}$.**

| Model | Metrics | | $D_{WT-2tabs-test}$ | $D_{WT-4tabs-test}$ | $D_{WT-6tabs-test}$ |
|---|---|---|---|---|---|
| Adaptive-BAPM | Previous | Acc | [68.7, 43.3] | [78.7, 47.7, 46.0, 46.5] | [81.0, 49.9, 48.2, 47.2, 48.5, 50.4] |
| | | Pre | [74.1, 14.0] | [75.3, 30.0, 8.9, 7.8] | [78.3, 22.6, 12.3, 6.3, 7.8, 16.8] |
| | | Rec | [61.4, 5.2] | [60.0, 8.2, 3.8, 3.1] | [60.3, 7.8, 4.4, 3.1, 3.1,5.2] |
| | Overall Basic | Acc | 21.6 | 21.9 | 16.7 |
| | | Pre | 52.3 | 52.1 | 57.6 |
| | | Rec | 37.6 | 23.4 | 17.2 |
| | Overall Advanced | Acc | 15.9 | 16.0 | 13.1 |
| | | Pre | 47.5 | 41.0 | 47.7 |
| | | Rec | 34.1 | 19.4 | 14.4 |
| **TMWF** | Previous | Acc | **[92.1, 62.8]** | **[95.5, 78.2, 45.5, 33.2]** | **[96.5, 89.8, 83.6, 75.9, 74.9, 75.7]** |
| | | Pre | **[91.0, 73.0]** | **[94.2, 84.7, 27.7, 7.3]** | **[93.7, 87.3, 80.8, 71.9, 67.8, 72.3]** |
| | | Rec | **[85.5, 41.4]** | **[92.3, 63.5, 22.9, 7.0]** | **[91.8, 86.5, 75.6, 65.2, 58.3, 62.9]** |
| | Overall Basic | Acc | **64.2** | **74.9** | **75.5** |
| | | Pre | **80.9** | **84.0** | **88.9** |
| | | Rec | **77.4** | **83.9** | **78.4** |
| | Overall Advanced | Acc | **35.0** | **29.6** | **68.2** |
| | | Pre | **49.2** | **38.5** | **81.6** |
| | | Rec | **60.5** | **46.7** | **74.7** |

**Table 3: TMWF ablation experiment results.**

| Model | Overall Basic | | | Overall Advanced | | |
|---|---|---|---|---|---|---|
| | Acc | Pre | Rec | Acc | Pre | Rec |
| Add Timeinfo | 73.8 | 89.4 | 76.6 | 68.7 | 84.4 | 74.3 |
| Original Model | 75.5 | 88.9 | 78.4 | 68.2 | 81.6 | 74.7 |
| No DF | 69.0 | 82.1 | 75.1 | 60.8 | 77.1 | 67.9 |
| No Transformer | 38.1 | 45.8 | 65.8 | 8.0 | 11.2 | 14.7 |

**Table 4: Evaluation results using Previous metrics based on dataset $D_{BAPM}$.**

| Model | 1st Page | | | 2nd Page | | |
|---|---|---|---|---|---|---|
| | Acc | Pre | Rec | Acc | Pre | Rec |
| BAPM | 93.2 | 93.5 | 92.9 | 82.9 | 84.9 | 83.2 |
| Adaptive-BAPM | 95.9 | 94.3 | 93.9 | 93.1 | 91.8 | 91.2 |
| **TMWF** | **97.7** | **95.9** | **95.7** | **97.4** | **95.5** | **95.5** |

We have conducted experiments to compare the performance of TMWF with and without the additional input of timing information, where the latter is represented by the "Add time info" condition in Table 3. Surprisingly, we found that timing sequences did little to improve model performance. Moreover, adding new features as input substantially slowed down the training process. Hence, we did not use timing information in subsequent experiments. More details on our exploration of timing information can be found in Appendix G.

In addition, we observed that the DF feature extractor in TMWF can extract more valuable information from the input sequence than the CNN-based feature extractor used in BAPM, as demonstrated by the performance gap between the "Original model" and "No DF" conditions in Table 3.

The most significant performance drop in Table 3 occurs when the Transformer module is removed from the original TMWF architecture. In this case, TMWF becomes a multi-tab classification model that directly classifies complete traces, including overlapping segments, making it difficult to distinguish specific website traces from mixed features of multiple pages. This phenomenon also indicates that the Transformer enables each tab query to focus on the trace area specific to a particular webpage, allowing the model to identify specific website traces from multi-tab traces. In this process, the Transformer promotes the model's ability to differentiate between features of different pages through its global modeling capability. Combining the experimental results in Table 2, it can be concluded that, in the same task scenario, the Transformer architecture is more robust to overlapping traces than the combination of the block partitioning module and self-attention module in BAPM.

## 6.3 Closed-world experiment

We conducted an evaluation on the real closed-world 2-tab trace dataset, $D_{BAPM}$, as provided by the work in [42]. The same evaluation metrics employed in the original paper were used: accuracy (Acc), precision (Pre), and recall (Rec) were calculated separately for each predicted page. The corresponding results are presented in Table 4.

We directly use the data from the research by [42] for the evaluation of BAPM in the table. It can be observed that Adaptive-BAPM not only does not degrade the performance compared to BAPM but also outperforms the original results in the paper. We attribute this phenomenon to the adoption of more reasonable parameter settings in our approach. Specifically, we adjust the length of each page in the input sequence from the original 4096 to 5120 while maintaining the number of blocks, and correspondingly adjusting

the block length from 128 to 160. We chose the value of 5120 because it is closer to the third quartile of the length of all single-tab traces in the current dataset, which enables us to retain more trace information.

TMWF achieved the best performance in all evaluation metrics, indicating its superior ability to effectively utilize overlapping regions. Thanks to this advantage, TMWF did not suffer significant performance degradation in classifying the second page. This phenomenon suggests that TMWF is more robust to cases where the head of the trace participates in the overlap, as previous research [36, 42] has shown that the head of the trace plays a more important role in classification.

## 6.4 Open-world experiment

In this section, we conduct open-world experiments on our collected $D_{CHR}$ dataset (for results based on $D_{TBB}$, refer to Appendix H). Building upon the exploration in Appendix E regarding the discrepancies between different synthesis methods, we opt for $D_{CHR-6tabs-ratio-train-40k}$ as the training set. This selection stems from the optimal model performance achieved with $M_{ratio}$ and its corresponding dataset scale for 2-tab traces. These multi-tab traces are synthesized using $M_{ratio}$ from a $\frac{4}{5}$ subset of our collected Chrome browser single-tab trace dataset.

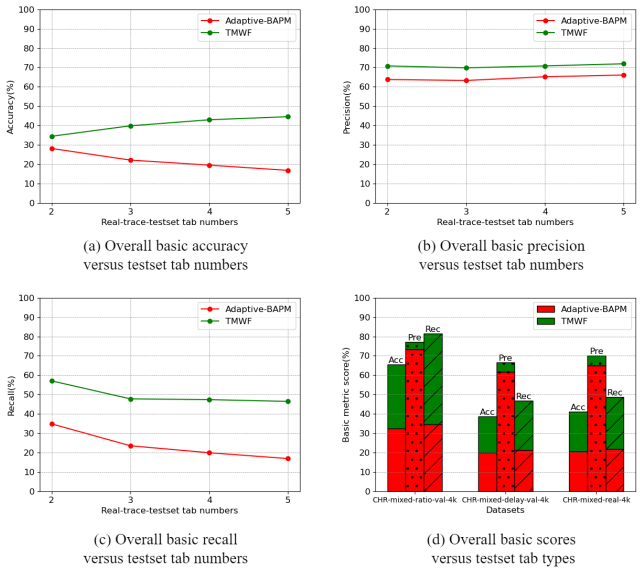
During the model evaluation phase, predictions are made on validation sets created using different synthesis methods and our collected real multi-tab trace test sets (each multi-tab trace test set with varying page numbers consists of 1000 samples). This approach facilitates a comparison of the disparities in model performance between synthesized traces and real traces. Aligning with the practices of [42] and [14], only packet direction sequences are utilized as the model's input. The classification results are depicted in Figure 5.

In dataset names, the **"mixed"** suffix denotes a composite of multi-tab trace datasets with equal page numbers of 2, 3, 4, and 5, respectively. For example, $D_{CHR-mixed-ratio-val-4k}$ signifies the synthesis of 2-tab, 3-tab, 4-tab, and 5-tab traces, each with 1000 instances, generated with $M_{ratio}$ from a $\frac{1}{5}$ subset of the Chrome browser single-tab trace dataset, followed by amalgamating these traces.

TMWF's performance still surpasses that of BAPM. For TMWF, contrasting the first cluster of metric scores in Figure 5-(d) with the others reveals an evident performance gap. Compared to precision, the degradation in accuracy and recall for TMWF is more pronounced, suggesting a higher proportion of false negatives among the model's erroneous judgments.

Regarding the trend of accuracy increasing with the number of multi-tab trace pages in Figure 5-(a), we conjecture that this phenomenon arises from the model's predicted results. Specifically, the false positive predictions, exceeding the number of pages contained in the ground truth, increase as the number of pages in the multi-tab trace decreases. Consequently, this leads to a decrease in Basic Accuracy. Despite the noticeable fluctuation in Accuracy, we observe relative stability in Basic Precision and Basic Recall in Figure 5-(b), (c).

Even with the insights from the experiments in Appendix E indicating that using $M_{ratio}$ as the synthesis method for training




(a) Overall basic accuracy versus testset tab numbers

(b) Overall basic precision versus testset tab numbers

(c) Overall basic recall versus testset tab numbers

(d) Overall basic scores versus testset tab types

**Figure 5: Model classification results on multi-tab trace validation sets and real multi-tab trace test sets using $D_{CHR-6tabs-ratio-train-40k}$ for Training.**
**(a), (b), and (c) depict variations in the evaluation results of the Basic series metrics as the number of real trace tabs increases. (d) showcases the model's performance across different types of mixed test sets (evaluated using the Basic series metrics).**

multi-tab traces is already an optimal solution in the absence of real traces, a discernible performance degradation is still evident when comparing the scores of the first cluster in Figure 5-(d) with the other clusters.

Furthermore, in comparison to TMWF's comprehensive performance on $D_{CHR-mixed-delay-val-4k}$ and $D_{CHR-mixed-real-4k}$, it is notable that although the training set generated using the $M_{delay}$ synthesis method does not effectively enhance the model's generalization ability and robustness (as shown in Appendix E), the model's performance on validation sets synthesized with this method mirrors the level seen in real multi-tab trace test sets with similar page structures.

In conclusion, we maintain that TMWF still struggles to achieve high accuracy in identifying real multi-tab traces, as the aforementioned performance levels are attained under controlled laboratory conditions. The base rate of popular monitored websites' homepages fails to reflect the genuinely sensitive pages of concern to potential attackers. Moreover, the results from the experiments in Appendix H demonstrate that the security strategy of the Tor browser implements a higher resistance against WF attacks. Nevertheless, it still poses certain threats to user anonymity under constrained circumstances.

**Confusion matrix.** To investigate the reasons for classification errors on the real-world dataset, we aggregated all real-world datasets using different page settings into one dataset (referred to as $D_{CHR-mixed-real-4k}$ in Figure 5-(d)), and analyzed TMWF's prediction results on this dataset. We plotted the prediction results of the ten classes with the highest error rates in a confusion matrix, as

shown in Figure 6. We also combined the remaining 40 monitored website classes from the 50 monitored websites into one class for display in the matrix, with the "no-tab" class as the last class in the confusion matrix.

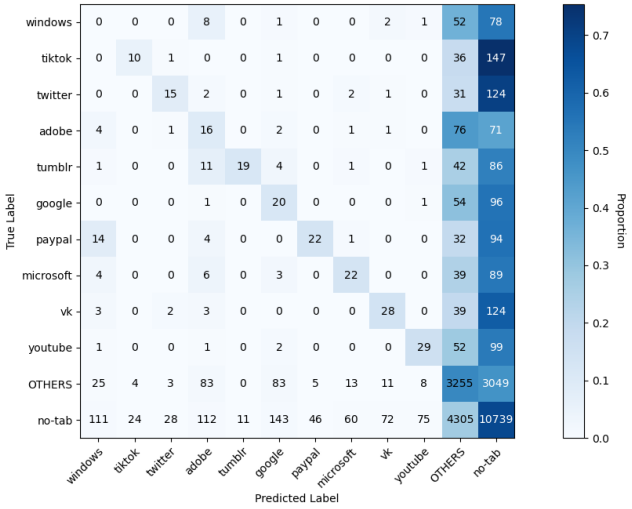



**Figure 6: Confusion matrix of predictions on the real multi-tab trace test set $D_{CHR-mixed-real-4k}$.**

From Figure 6, it is apparent that the majority of the erroneous predictions classify monitored websites as unmonitored ones (i.e., "no-tab"), with a few predictions misclassifying monitored websites as other unmonitored websites. Additionally, TMWF reports a high accuracy for the no-tab category, which aligns with our expectations.

## 7 DISCUSSION

In this section, we will discuss some of the limitations of TMWF and existing multi-tab WF attacks.

The recognition of WF for completely overlapping trace segments remains an unresolved issue. While our work has improved the effectiveness of feature mining for overlapping segments, our approach still relies mainly on pure segments. When users access multiple web pages simultaneously in a parallel manner, the packet sequences of multiple pages will be completely mixed together. Compared to the assumption of serial access currently used in WF attacks, the completely mixed traffic traces will magnify the confusion effect of their features, thus affecting model performance.

Although this paper does not evaluate TMWF's performance on traffic traces using WF defense techniques, its performance in this scenario will be greatly compromised. Compared to single-tab WF attack models, it is more difficult for multi-tab WF attack models to extract valuable high-level representations from the raw input. Because WF defense techniques [22, 35, 38] further undermine the integrity of pure segments in multi-tab traces, the performance of multi-tab WF attack models will be more severely impacted. Although our collected real-world multi-tab trace dataset extends the website scope to the open-world, all current real-world multi-tab trace datasets still cannot fully simulate users' real browsing behavior. For example, manual operations such as clicking links and filling out forms can affect the generation of real-world traces. Additionally, the real-world multi-tab trace dataset only breaks the single-tab assumption, and it is still limited by a series of assumption conditions, such as webpages under the same domain with different subdomains, Tor version differences, user network environment differences, website base rate, etc.

## 8 CONCLUSION

This paper proposes a deep learning detection model called TMWF based on the Transformer architecture for multi-tab WF attacks with overlapping webpage traces. The model uses DFNet as a feature extractor, utilizes the self-attention mechanism inside the Transformer for global modeling, and employs our proposed new evaluation metric for performance evaluation. The study shows that TMWF outperforms existing methods and does not rely on manually designed features. Equipped with our proposed new training method, it adapts to identifying individual page traces from mixed multi-tab traces. Additionally, this paper collects a real open-world multi-tab trace dataset that includes unmonitored website traces. These contributions provide new ideas for multi-tab WF attacks and have important research significance and application value.

## ACKNOWLEDGMENTS

We thank the anonymous reviewers for their feedback and helpful suggestions to improve the paper. This work is supported by the National Natural Science Foundation of China (No.62162060).

## REFERENCES

[1] Andreas Zinnen Andriy Panchenko, Lukas Niessen. 2011. Fingerprinting Websites Using Traffic Analysis. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society, WPES 2011, Chicago, IL, USA, October 17, 2011.* ACM, 103–114.

[2] Jan Pennekamp Andriy Panchenko, Fabian Lanze. 2016. Website Fingerprinting at Internet Scale. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016.* The Internet Society.

[3] Niki Parmar Ashish Vaswani, Noam Shazeer. 2017. Attention is All You Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA.* 5998–6008.

[4] Giovanni Cherubin, Rob Jansen, and Carmela Troncoso. 2022. Online website fingerprinting: Evaluating website fingerprinting attacks on Tor in the real world. In *31st USENIX Security Symposium (USENIX Security 22).* 753–770.

[5] Manfred Hauswirth Christian von der Weth. 2013. DOBBS: Towards a Comprehensive Dataset to Study the Browsing Behavior of Online Users. In *2013 IEEE/WIC/ACM International Conferences on Web Intelligence, WI 2013, Atlanta, GA, USA, November 17-20, 2013.* IEEE Computer Society, 51–56.

[6] Guodong Huang, Chuan Ma, Ming Ding, Yuwen Qian, Chunpeng Ge, Liming Fang, and Zhe Liu. 2023. Efficient and Low Overhead Website Fingerprinting Attacks and Defenses Based on TCP/IP Traffic. In *Proceedings of the ACM Web Conference 2023* (Austin, TX, USA) *(WWW '23).* Association for Computing Machinery, New York, NY, USA, 1991–1999. https://doi.org/10.1145/3543507.3583200

[7] Rob Jansen and Ryan Wails. 2023. Data-Explainable Website Fingerprinting with Network Simulation. *Proceedings on Privacy Enhancing Technologies* 4 (2023), 559–577.

[8] Ross B. Girshick Joseph Redmon, Santosh Kumar Divvala. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016.* IEEE Computer Society, 779–788.

[9] Shaoqing Ren Kaiming He, Xiangyu Zhang. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016.* IEEE Computer Society, 770–778.

[10] Gunes Acar Marc Juárez, Sadia Afroz. 2014. A Critical Evaluation of Website Fingerprinting Attacks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014.* ACM, 263–274.

[11] Nate Mathews, James K Holland, Se Eun Oh, Mohammad Saidur Rahman, Nicholas Hopper, and Matthew Wright. 2023. SoK: A critical evaluation of efficient website fingerprinting defenses. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 969–986.
[12] Gabriel Synnaeve Nicolas Carion, Francisco Massa. 2020. End-to-End Object Detection with Transformers. In *ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*. Springer, 213–229.
[13] Se Eun Oh, Nate Mathews, Mohammad Saidur Rahman, Matthew Wright, and Nicholas Hopper. 2021. GANDaLF: GAN for data-limited fingerprinting. *Proceedings on Privacy Enhancing Technologies* 2021, 2 (2021).
[14] Marc Juárez Payap Sirinam, Mohsen Imani. 2018. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. ACM, 1928–1943.
[15] Mohammad Saidur Rahman Payap Sirinam, Nate Mathews. 2019. Triplet Fingerprinting: More Practical and Portable Website Fingerprinting with N-shot Learning. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. ACM, 1131–1148.
[16] Tobias Pulls and Rasmus Dahlberg. 2020. Website Fingerprinting with Website Oracles. *Proc. Priv. Enhancing Technol.* 2020, 1 (2020), 235–255.
[17] Jian Qu, Xiaobo Ma, Jianfeng Li, Xiapu Luo, Lei Xue, Junjie Zhang, Zhenhua Li, Li Feng, and Xiaohong Guan. 2023. An Input-Agnostic Hierarchical Deep Learning Framework for Traffic Fingerprinting. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 589–606. https://www.usenix.org/conference/usenixsecurity23/presentation/qu
[18] Paul F. Syverson Roger Dingledine. 2002. Fingerprinting Websites Using Traffic Analysis. In *Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14-15, 2002, Revised Papers*. Springer, 171–178.
[19] Paul F. Syverson Roger Dingledine, Nick Mathewson. 2004. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*. USENIX, 303–320.
[20] Mohammad Saidur Rahman, Payap Sirinam, Nate Mathews, Kantha Girish Gangadhara, and Matthew Wright. 2019. Tik-Tok: The Utility of Packet Timing in Website Fingerprinting Attacks. *arXiv e-prints* (2019), arXiv–1902.
[21] Albert Kwon Sanjit Bhat, David Lu. 2019. Var-CNN: A Data-Efficient Website Fingerprinting Attack Based on Deep Learning. *Proc. Priv. Enhancing Technol.* (2019), 292–310.
[22] Pablo Serrano Sébastien Henri, Gines Garcia-Aviles. 2020. Protecting against Website Fingerprinting with Multihoming. *Proc. Priv. Enhancing Technol.* 2020, 2 (2020), 89–110.
[23] Ross B. Girshick Shaoqing Ren, Kaiming He. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. 91–99.
[24] Meng Shen, Kexin Ji, Zhenbo Gao, Qi Li, Liehuang Zhu, and Ke Xu. 2023. Subverting Website Fingerprinting Defenses with Robust Traffic Representation. In *32nd USENIX Security Symposium (USENIX Security 23)*. 607–624.
[25] Ian Goldberg Tao Wang. 2013. Improved Website Fingerprinting on Tor. In *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013, Berlin, Germany, November 4, 2013*. ACM, 201–212.
[26] Ian Goldberg Tao Wang. 2016. On Realistically Attacking Tor with Website Fingerprinting. In *Proc. Priv. Enhancing Technol.* 21–36.
[27] Ian Goldberg Tao Wang. 2017. Walkie-Talkie: An Efficient Defense Against Passive Website Fingerprinting Attacks. In *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*. USENIX Association, 1375–1390.
[28] Rishab Nithyanand Tao Wang, Xiang Cai. 2014. Effective Attacks and Provable Defenses for Website Fingerprinting. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*. USENIX, 143–157.
[29] Alexander Veicht, Cedric Renggli, and Diogo Barradas. 2022. DeepSE-WF: Unified Security Estimation for Website Fingerprinting Defenses.
[30] Marc Juárez Vera Rimmer, Davy Preuveneers. 2018. Automated Website Fingerprinting through Deep Learning. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society.
[31] Tom van Goethem Vera Rimmer, Theodor Schnitzler. 2022. Trace Oddity: Methodologies for Data-Driven Traffic Analysis on Tor. *Proc. Priv. Enhancing Technol.* 2022 (2022), 314–335.
[32] Samaneh Tajalizadehkhoob Victor Le Pochat, Tom van Goethem. 2019. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society.
[33] Tao Wang. 2020. Designing a Better Browser for Tor with BLAST.. In *NDSS*.
[34] Tao Wang. 2020. High precision open-world website fingerprinting. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 152–167.
[35] Tao Wang. 2021. The One-Page Setting: A Higher Standard for Evaluating Website Fingerprinting Defenses. In *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*. ACM, 2794–2806.
[36] Eric Chan-Tin Weiqi Cui, Tao Chen. 2020. More Realistic Website Fingerprinting Using Deep Learning. In *40th IEEE International Conference on Distributed Computing Systems, ICDCS 2020, Singapore, November 29 - December 1, 2020*. IEEE, 333–343.
[37] Ethan Witwer, James K Holland, and Nicholas Hopper. 2022. Padding-only defenses add delay in Tor. In *Proceedings of the 21st Workshop on Privacy in the Electronic Society*. 29–33.
[38] Jens Hiller Wladimir De la Cadena, Asya Mitseva. 2020. TrafficSliver: Fighting Website Fingerprinting Attacks with Traffic Splitting. In *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*. ACM, 1971–1985.
[39] Zhenhao Guo Yanbin Wang, Haitao Xu. 2022. snWF: Website Fingerprinting Attack by Ensembling the Snapshot of Deep Learning. *IEEE Trans. Inf. Forensics Secur.* 17 (2022), 1214–1226.
[40] Qilei Yin, Zhuotao Liu, Qi Li, Tao Wang, Qian Wang, Chao Shen, and Yixiao Xu. 2021. An automated multi-tab website fingerprinting attack. *IEEE Transactions on Dependable and Secure Computing* 19, 6 (2021), 3656–3670.
[41] Yue Cao Ze Liu, Yutong Lin. 2021. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 9992–10002.
[42] Gaopeng Gou Zhong Guan, Gang Xiong. 2021. BAPM: Block Attention Profiling Model for Multi-tab Website Fingerprinting Attacks on Tor. In *ACSAC '21: Annual Computer Security Applications Conference, Virtual Event, USA, December 6 - 10, 2021*. ACM, 248–259.

# A TMWF ARCHITECTURE DETAILS

TMWF consists of three main stages. The backbone module is responsible for extracting high-level feature representations from the raw input, which serves as the foundation for subsequent processes. The Transformer architecture is responsible for extracting WF embeddings for multiple pages from the contextual feature sequence, and the classification head then classifies multiple WF embeddings and outputs label predictions.

**Backbone.** In TMWF, we utilize DFNet to perform local modeling on the input multi-tab traces. Specifically, we remove the feed-forward network used for classification in DFNet and retain the rest of the original structure to transform the input multi-tab traces into a feature map. This feature map is equivalent to a sequence of WF features, where each element in the sequence is a feature embedding (a multi-dimensional vector), and each embedding contains some spatial information of the original traces. Since the convolutional kernels in DFNet calculate the feature embeddings one by one using a sliding window mechanism, the feature map output by DFNet also has positional relationships between the front and back. In other words, in the feature map sequence, the feature embeddings located at the front more fully reflect the front of the original traces, and the same is true for the feature embeddings at the back.

Although the feature embeddings output by the backbone carries positional relationships, the Transformer lacks inductive bias towards positional information in its design and cannot learn positional information from input features. To compensate for this deficiency, we add learnable positional encoding to the feature sequence outputted by the backbone and use the sum as the input to the Transformer encoders. Assuming that the two-dimensional feature map $F$ output by the backbone has $l$ feature embeddings, with a dimensionality of $d$, represented by $F = [\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_l] \in R^{l \times d}$, where each element $\mathbf{f}_i$ is a vector of dimensionality $d$. The shape of the positional encoding $P$ is exactly the same as that of $F$, i.e.,

$P = [\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_l] \in R^{l \times d}$, where $\mathbf{p}_i$ is also a vector of dimensionality $d$. We use a projection layer to add $P$ and $F$ and denote the input to the encoders as $X$, which is

$$X = LayerNorm(FW + \mathbf{b}) + P, X \in R^{l \times d} \quad (12)$$

$LayerNorm$ represents layer normalization, and $W \in R^{d \times d}$, $\mathbf{b} \in R^d$ are the parameters required for model operations. During the gradient update process of the model, the parameters of the positional encoding will be continuously updated to adapt to all WF features.

**Transformer Encoders.** The vanilla Transformer encoder consists of a multi-head attention mechanism and a feedforward network, where the multi-head attention mechanism is responsible for global modeling and the feedforward network is responsible for feature selection. Suppose the input to the encoder is $X \in R^{l \times d}$. We denote the intermediate and final outputs of the multi-head attention mechanism as $O$ and $O^{'}$, respectively. The computation process can be expressed using the following equations:

$$O = MultiHeadAttention(X) = Concat(head_1, \cdots, head_h)W_O \quad (13)$$

$$head_t = Softmax(\frac{XW_Q^t(XW_K^t)^T}{\sqrt{d/h}})(XW_V^t), t = 1, 2, \cdots, h \quad (14)$$

$$O^{'} = LayerNorm(O + X), O^{'} \in R^{l \times d} \quad (15)$$

The operation process of the context embedding sequence $Z$ output by the feedforward network is as follows:

$$Z = LayerNorm(O^{'} + (max(0, O^{'}W_1 + \mathbf{b}_1))W_2 + \mathbf{b}_2), Z \in R^{l \times d} \quad (16)$$

Where $Concat$ represents the concatenation function, and $W_Q^t \in R^{d \times \frac{d}{h}}$, $W_K^t \in R^{d \times \frac{d}{h}}$, $W_V^t \in R^{d \times \frac{d}{h}}$, $W_O \in R^{d \times d}$, $W_1 \in R^{d \times 4d}$, $W_2 \in R^{4d \times d}$, $\mathbf{b}_1 \in R^{4d}$, $\mathbf{b}_2 \in R^d$ are the parameters required for model operations. $h$ is the number of attention heads we set. The above operation process can be summarized as:

$$Z = TransformerEncoder(X) \quad (17)$$

We use $N_E$ encoders to model the normalized feature sequence to extract the WF context feature sequence. The encoders can extract higher-level global features from the feature sequence output by the backbone, which helps to distinguish WF from different pages.

**Tab Queries.** Although the Transformer encoder is effective in global modeling, its output context feature sequence does not separate the features of different pages in multi-tab traces. To this end, we introduce $N$ tab queries (corresponding to object queries in DETR) as references for the Transformer decoders to query the fingerprint embeddings of different pages. Each tab query is essentially a learnable positional encoding. However, unlike the positional encoding used in the description mentioned above, the tab query learns spatial information specific to independent pages. In other words, we expect the $i$-th tab query to learn the representation of spatial information for the $i$-th page in all original traces. We use $T_Q$ to denote the tab queries, where $N$ is the number of tab queries we set. This can be represented as $T_Q = [\mathbf{tq}_1, \mathbf{tq}_2, \cdots, \mathbf{tq}_N] \in R^{N \times d}$.

**Transformer Decoders.** We use $N_D$ vanilla Transformer decoders to extract the WF embeddings of $N$ pages from the contextual feature sequence output by the encoders. The operation process of each decoder is described in detail next. Suppose the input to the decoder is contextual feature sequence $Z \in R^{l \times d}$. The decoder first employs a multi-head attention mechanism to allow for information exchange between tab queries, facilitating the model's understanding of relevant information in the data. This process can be represented as

$$T_Q^{'} = MultiHeadAttention(T_Q) \quad (18)$$

Next, the tab queries that have undergone interaction are combined with the contextual feature sequence $Z$ as inputs to the cross-attention mechanism. The mechanism can find a plausible alignment between tab queries and contextual feature sequence so that each tab query finds the fingerprinting feature of the page to which it belongs. We use $O$ and $O^{'}$ to represent the intermediate and final output results, respectively. The computation process is as follows:

$$O = CrossAttention(T_Q^{'}, Z) = Concat(head_1, \cdots, head_h)W_O \quad (19)$$

$$head_t = Softmax(\frac{(T_Q^{'}W_Q^t)(ZW_K^t)^T}{\sqrt{d/h}})(ZW_V^t), t = 1, 2, \cdots, h \quad (20)$$

$$O^{'} = LayerNorm(O + T_Q^{'}), O^{'} \in R^{N \times d} \quad (21)$$

The WF features of each page (represented as $\mathbf{e}_i$) is subjected to feature selection by a feed-forward network, as shown in the following equation:

$$E = LayerNorm(O^{'} + (max(0, O^{'}W_3 + \mathbf{b}_3))W_4 + \mathbf{b}_4), E \in R^{N \times d} \quad (22)$$

$$E = [\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_N], E \in R^{N \times d} \quad (23)$$

Here, $W_Q^t \in R^{d \times \frac{d}{h}}$, $W_K^t \in R^{d \times \frac{d}{h}}$, $W_V^t \in R^{d \times \frac{d}{h}}$, $W_O \in R^{d \times d}$, $W_3 \in R^{d \times 4d}$, $W_4 \in R^{4d \times d}$, $\mathbf{b}_3 \in R^{4d}$, $\mathbf{b}_4 \in R^d$ are the parameters required for model operations. $h$ is the number of attention heads we set. The above operation process can be summarized as:

$$E = TransformerDecoder(T_Q, Z) \quad (24)$$

$N_D$ decoders extracted WF embeddings $E_{WF} \in R^{N \times d}$ from the contextual feature sequence output by the encoders for subsequent WF recognition tasks.

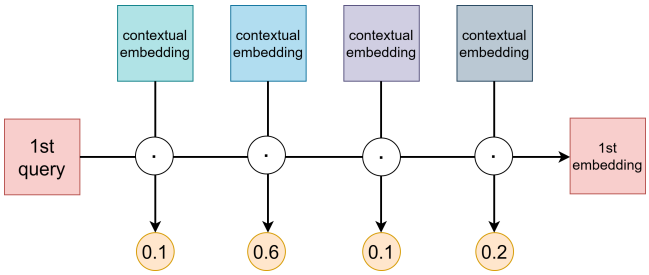



**Figure 7: An example of aligning tab queries with the contextual feature sequence.**

Figure 7 demonstrates an example of computing a WF embedding. Assuming that Transformer has extracted four context embeddings, after vector inner product operation, we obtain the similarity weights of the tab query for the first page with the four context embeddings, which are 0.1, 0.6, 0.1, and 0.2 (normalized by $Softmax$). We then weigh and sum each of the four context embeddings based

**Table 5: Evaluation results of TMWF on $D_{WT-6tabs-test}$ with different parameter combinations.**
**Experimental setup: GPU: NVIDIA RTX4080 mobile with 12GB VRAM; RAM: 32GB.**

| Parameter Combinations | Training Time | Basic Accuracy | Basic Precision | Basic Recall |
|---|---|---|---|---|
| (1,1,2) | 100min | 72.3 | 88.4 | 75.1 |
| (1,1,4) | 100min | 74.2 | 89.0 | 77.2 |
| (1,1,8) | 100min | 74.3 | 89.1 | 77.1 |
| **(2,2,8)** | 120min | **75.5** | 88.9 | **78.4** |
| (2,2,16) | 120min | 75.0 | 88.8 | 78.0 |
| (3,3,8) | 150min | 74.8 | **89.3** | 77.7 |
| (3,3,16) | 150min | 74.5 | 86.5 | **78.4** |

on their corresponding weights to obtain the WF embedding for the first page.

**Classification.** We use a classification head to classify the WF embeddings of $N$ pages extracted by the Transformer decoders. This classification head consists of a fully connected layer and a Softmax function. We apply the same classification head to all page embeddings, which converts each page embedding into a probability distribution over websites, and the website with the highest probability is considered the predicted result of TMWF for that page. Assuming there are $C$ websites in total, the above process can be described as:

$$Pr = Softmax(E_{WF}W_5 + \mathbf{b}_5) = [\mathbf{pr}_1, \mathbf{pr}_2, \cdots, \mathbf{pr}_N], Pr \in R^{N\times C} \tag{25}$$

Where $\mathbf{pr}_i$ denote the probability distribution over websites for the $i$-th page. While TMWF produces $N$ website prediction results, we expect that the number of pages in the actual prediction results of the model is typically no more than $N$. For non-monitored websites and redundant prediction results, we uniformly assign the label "**no-tab**".

## B NEW METRICS' INTUITION

To aid in illustrating the intuition behind our newly proposed metrics, we envisage an evaluation scenario where a TMWF with $N = 4$ is applied to predict a test set comprising five 2-tab traces, as depicted in Figure 8.

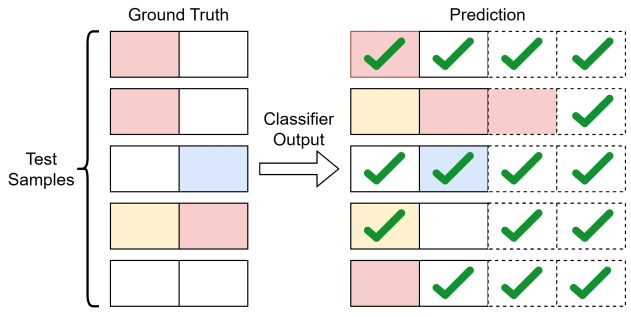



**Figure 8: An example of a test set containing five 2-tab trace samples.**

On the left side of Figure 8, we have the ground truth of the dataset samples, while on the right side, we display the model's fixed number of prediction results. The ticked areas signify accurate prediction results. Each box corresponds to a segment of a website's trace. The labels in the white-filled boxes are denoted as "no-tab", whereas the labels in the colored boxes represent various websites within the monitored set.

Using accuracy as an example, a straightforward approach would be to calculate accuracy solely based on the prediction results derived from the unfilled multi-tab traces, indicated by the solid boxes in Figure 8. However, this method would overlook whether the additional prediction results generated by TMWF are correct. Conversely, directly computing accuracy based on all prediction results would result in an inflated accuracy due to TMWF's high performance in recognizing redundant predictions.

The computed results for our proposed Overall series evaluation metrics are as follows:

$$Accuracy_{Basic} = \frac{1+1+1+1+0}{1+2+1+2+1} = \frac{4}{7} \tag{26}$$

$$Accuracy_{Advanced} = \frac{1+0+1+1+0}{1+3+1+2+1} = \frac{3}{8} \tag{27}$$

In conclusion, the newly introduced metrics accurately reflect the genuine performance of the model in identifying monitored websites. Moreover, by labeling non-monitored websites as "no-tab", these metrics can be extended to other WF models.

## C PARAMETER TUNING

As depicted in Figure 3, within TMWF, we modified only the original input dimension (i.e., the length of the trace sequence) due to our utilization of the default parameters of the backbone module (DFNet [14]). Consequently, we focused our parameter-tuning efforts on the Transformer module. For this module, after excluding parameters determined by other components, we identified three customizable parameters: the number of encoders and decoders within the Transformer architecture, as well as the number of attention heads. These parameter combinations are denoted as **"(encoder number, decoder number, attention head number)"**. For each combination, training was conducted on $D_{WT-6tabs-train}$, with subsequent testing on $D_{WT-6tabs-test}$ (containing approximately 22,000 and 5,500 samples respectively).

Additionally, our model employed the Adam optimizer coupled with the PyTorch default cross-entropy loss function for training. We set the batch size to 80, the number of epochs to 50, and the learning rate to 0.0005. Based on these configurations, our experiments indicated that the model's performance on the validation set converged between the 20th and 30th epochs. We explored seven sets of parameter combinations, culminating in evaluation results (based on our proposed new metrics) as presented in Table 5.

Clearly, when configuring the Transformer architecture with two encoders and two decoders, and utilizing eight attention heads, TMWF achieves a favorable balance between performance and training time on the current 6-tab dataset. In conclusion, our model is considered a lightweight structure that demonstrates commendable performance within the range of parameters we have experimented with.

**Table 6: Performance of the model on datasets synthesized by different methods.**

| Guiding Principles | 1st Page | | | 2nd Page | | |
|---|---|---|---|---|---|---|
| | Acc | Pre | Rec | Acc | Pre | Rec |
| Packet Sequence | 82.8 | 83.9 | 82.7 | 63.6 | 64.3 | 63.1 |
| Loading Period | 80.3 | 82.4 | 80.4 | 63.2 | 66.2 | 63.2 |

# D MULTI-TAB TRACES SYNTHESIS METHOD

When synthesizing multi-tab traces, it is required that each packet in the single-tab traces used for synthesis contains a timing feature, such as a timestamp or time interval. This is necessary to maintain the timing feature of the original sequence when mixing 2-tab traces at arbitrary overlap ratios. This forms the cornerstone of the model's ability to distinguish different page sequences from unlabeled sequences. Take the example of synthesizing a 2-tab trace based on two single-tab traces, as shown in Figure 2.

After preprocessing, all packet timestamps are processed into values starting from 0 to represent the loading time of the current webpage. When selecting a certain overlap ratio, we calculate the starting point of the overlap segment of the first page A according to this ratio and obtain the timestamp of the corresponding packet at that position. We then increase the timestamps of all packets on the second page by this value and sort the packets of these two pages according to their reset timestamps.

We have observed that different works at multi-tab WF employ different guiding principles when synthesizing traces. One method [36] employs the length of the data packet sequence as a guide, while another [42] uses the time span of the loading process. Due to the non-uniform distribution of data packets (Tor cells) generated during webpage loading, the synthesized multi-tab WF resulting from different guiding principles may differ significantly even when using the same overlap ratio. For example, for a single-tab WF with data packets concentrated at the end of the time axis, when an overlap ratio of 10% of the loading time is selected, the proportion of data packets from that page that participate in the overlapping segment with another page in the final synthesized trace may exceed 50%.

We are concerned that using fixed time ratios to synthesize WF may lead to significant discrepancies in the number of data packets in the pure segments, and we thus attempted to synthesize multi-tab WF using various overlap ratios based on the period of the loading process. We found that when the period of the loading process was used as a guide, the distribution of the proportion of different data packet numbers in the final synthesized multi-tab WF became more even as the overlap ratio increased. This trend alleviated the impact of the abnormal phenomenon that we had speculated. Based on this finding, we randomly selected any overlap ratio between 0.1 and 0.5 to simulate diverse access scenarios in real-world situations. For the multi-tab WF dataset synthesized using the two guiding principles of employing the length of the data packet sequence and using the period of the loading process with various overlap ratios, our evaluation results on BAPM (as shown in Table 6) demonstrated that the performance difference between the two methods was negligible. Therefore, we chose to synthesize multi-tab WF based on the period of the loading process (i,e., $M_{ratio}$), as [42] does, in our subsequent experiments.

# E SYNTHETIC MULTI-TAB TRACES EXPLORATION

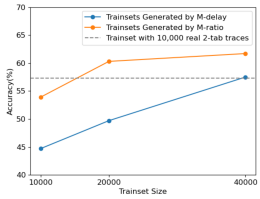
During the collection of real multi-tab traces, we observed that due to the unknown attributes of traces, it is not feasible to determine the overlap ratio between consecutive traces. Instead, we introduced random delays between consecutive page access behaviors to simulate trace overlaps. However, this approach introduces uncertainties. For instance, a page might not have overlapping segments if it finishes loading before the subsequent page is accessed. Also, a complete trace sequence of a certain page might be overshadowed by pages with longer loading times, resulting in no pure segments.

The existing methodologies for synthesizing multi-tab traces, as mentioned in Appendix D, rely on the overlap ratio. Consequently, datasets generated using these methodologies inherently contain both overlapping and pure segments. This means that the access patterns in datasets created using these methodologies differ from the access patterns observed during the collection of real multi-tab traces. Thus, the current overlap-based synthesis methods further exacerbate the dissimilarity between synthesized and real traces.

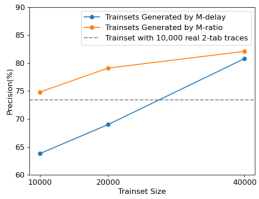
In light of this, we have devised a new synthesis method that aligns with the access patterns observed during the collection of real multi-tab traces. In this approach, we randomly select a value within a fixed time range (same as the range setting during the collection of real multi-tab traces) as the access delay between consecutive pages. When the number of accessed pages exceeds three, an additional delay is introduced between the second and subsequent pages to supplement pure segment length, as illustrated in Figure 4. In the following sections, we will refer to the overlap-based synthesis method as $M_{ratio}$, while our proposed approach will be referred to as $M_{delay}$.

Given the potential disparities between multi-tab traces collected from real-world environments and those manually synthesized, we conducted experiments using additional collected real 2-tab traces, notated as $D_{CHR-2tabs-real-10k}$. Additionally, we trained TMWF on 2-tab traces synthesized through various methods, including $D_{CHR-2tabs-ratio-train}$ and $D_{CHR-2tabs-delay-train}$ (while varying the size of the synthetic training set). Subsequently, we tested these models on the real 2-tab traces dataset, $D_{CHR-2tabs-real-1k}$, to assess the impact of these traces and the number of samples used for training on the model's performance. The experimental results are depicted in Figure 9.

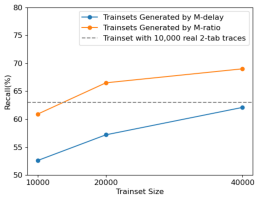
Based on the experimental findings depicted in Figure 9, we can draw the following conclusions: within the same dataset scale, the model's performance is the strongest when trained on real multi-tab traces, as the traffic patterns in the training set multi-tab traces closely resemble those in the test set. However, by augmenting the size of the synthetically generated multi-tab trace training set, the performance of TMWF can even surpass that of the original-sized real multi-tab trace training set. It's worth noting that all the synthetic multi-tab training sets we employed are generated from the original collected single-tab trace dataset (no additional subset division of the original dataset was performed here).




(a) Overall basic accuracy versus trainset sizes




(b) Overall basic precision versus trainset sizes




(c) Overall basic recall versus trainset sizes

**Figure 9: Comparison of evaluation results on the same test set $D_{CHR-2tabs-real-1k}$ by altering the training set utilized by TMWF.**
**The horizontal dashed lines parallel to the x-axis represent the evaluation results of TMWF trained on $D_{CHR-2tabs-real-10k}$.**

Furthermore, from Figure 9, we can also observe that, within the same training set scale, models trained on multi-tab trace training sets synthesized by $M_{ratio}$ consistently outperform those trained on sets synthesized by $M_{delay}$. This phenomenon is unexpected, considering that we introduced the new $M_{delay}$ synthesis method precisely to better emulate the access patterns when collecting real multi-tab traces. We attribute these results to the fact that multi-tab traces synthesized using $M_{ratio}$ possess more overlapping segments, enhancing the model's robustness.

As generating larger training sets demands more time, we did not explore dataset sizes beyond 40,000. Given our observation that the model achieved its best performance with $D_{CHR-2tabs-ratio-train-40k}$, we also employ a multi-tab trace dataset of this scale for model training in Section 6.4.

# F SUPPLEMENTARY MODEL VALIDATION EXPERIMENT

Due to the relatively earlier data collection time of the dataset from [27] as used in Section 6.1, it implies that the data utilized in the experiment might be outdated, and conducting experiments on more recent datasets (even if representing the same pages) could yield different results. Therefore, based on our newly collected $D_{TBB}$ dataset, we have conducted the same experiments (as described in Section 5.3) and evaluation process to further validate the model's performance.

We maintained the same strategy as described in Section 5.4 for "synthesizing multi-tab traces from single-tab traces from [27]". We synthesized multi-tab trace samples with random overlap ratios and website distributions using the single-tab traces we collected. Following the naming convention for $D_{TBB}$ outlined in Section 5.4, we describe the experiments in this section as follows: We trained Adaptive-BAPM and TMWF based on $D_{TBB-6tabs-ratio-train-20k}$ and then evaluated their performance using both the previous metrics and the new metrics proposed by us on $D_{TBB-2tabs-ratio-val-5k}$, $D_{TBB-4tabs-ratio-val-5k}$, and $D_{TBB-6tabs-ratio-val-5k}$, respectively. It is important to note that the aforementioned training and testing sets are derived from two segregated subsets of single-tab traces. The results of these experiments are presented in Table 7.

In reference to Table 7, we observed the same phenomena as described in Section 6.1, including the changes in metric scores and the performance gap between TMWF and Adaptive-BAPM. This suggests that TMWF is equally applicable to the latest Tor browser. Additionally, we noticed that, in comparison to the experimental results presented in Table 2, the evaluation scores on our collected dataset have generally and notably decreased. This indicates that the newer versions of the Tor browser possess stronger resistance against WF identification.

# G INPUT DATA'S EXPLORATION

Currently, most of the latest WF works [14, 15, 30, 39, 42] follow the method proposed in [25]. This involves extracting Tor cells from captured TCP packets, with +1 indicating the cell direction from the client to the target website, and -1 indicating the opposite direction. The resulting traffic traces are then converted into direction sequences and used as input for the classifier. In traditional single-tab WF scenarios, deep learning models using this type of input have already achieved outstanding classification performance, so researchers typically do not consider adding new raw input data to the classifier. However, in multi-tab WF scenarios, the input sequence of the overlapping parts of the pages is confused, leading to a degradation in model performance. Therefore, we propose reintroducing the timing features of the data packets and exploring their potential for improving the performance of multi-tab WF models. Additionally, in Section 5.4, we synthesized multi-tab traces manually, which unavoidably involved using data packet timestamps as dependencies. Therefore, we believe that attackers have a strong incentive to make full use of the information they need to collect.

We opted to modify the feature extractor used in TMWF to incorporate the timing features of Tor cells. Considering that network environment differences can significantly impact loading times, rendering the classifier unable to learn effective features, we processed the original timestamp sequence into a loading time interval sequence (the difference between each cell timestamp and the trace start time). We further standardized the sequence by dividing each cell's current loading time by the total loading time of the trace, as per the formula: (timestampc-timestamps)/(timestampe-timestamps). We speculate that this operation can mitigate the interference of network dynamics. Building upon this, we experimented with several ways to utilize the timing representation of the traces and found that incorporating timing representation as the second channel of input data yielded the best performance for the model.

We trained TMWF on dataset $D_{WT-6tabs-train}$, using only the direction sequence or both the direction and time sequences. We found that when DF was used as the feature extractor, the evaluation results of the two models were almost identical on the $D_{WT-6tabs-test}$. The evaluation results are shown in Table 8. However, the evaluation results on the $D_{WT-2tabs-test}$ and $D_{WT-4tabs-test}$ sets differed significantly: the TMWF with time sequence input

**Table 7: Evaluation of prediction on $D_{TBB-2tabs-ratio-val-5k}$, $D_{TBB-4tabs-ratio-val-5k}$, and $D_{TBB-6tabs-ratio-val-5k}$ based on models trained with $D_{TBB-6tabs-ratio-train-20k}$.**

| Model | Metrics | | $D_{TBB-2tabs-ratio-val-5k}$ | $D_{TBB-4tabs-ratio-val-5k}$ | $D_{TBB-6tabs-ratio-val-5k}$ |
|---|---|---|---|---|---|
| Adaptive-BAPM | Previous | Acc | [56.5, 44.9] | [67.5, 48.0, 45.1, 45.3] | [66.9, 51.8, 47.0, 45.8, 44.9, 48.2] |
| | | Pre | [60.0, 5.4] | [64.9, 8.9, 3.5, 1.7] | [62.0, 8.8, 6.7, 3.0, 7.5, 16.5] |
| | | Rec | [44.7, 2.5] | [53.2, 2.9, 2.6, 2.1] | [51.6, 3.3, 3.2, 3.0, 3.3, 6.3] |
| | Overall Basic | Acc | 17.2 | 15 | 11.6 |
| | | Pre | 47.6 | 50.8 | 54.1 |
| | | Rec | 24.7 | 15.1 | 11.2 |
| | Overall Advanced | Acc | 11.4 | 10.7 | 8.9 |
| | | Pre | **43.1** | **42.9** | 45.7 |
| | | Rec | 20.6 | 12.0 | 9.2 |
| **TMWF** | Previous | Acc | **[80.2, 51.6]** | **[83.4, 59.8, 40.3, 29.7]** | **[84.0, 69.5, 64.3, 62.4, 68.1, 80.4]** |
| | | Pre | **[84.2, 54.0]** | **[84.0, 63.0, 17.5, 2.2]** | **[83.9, 63.0, 54.2, 52.8, 59.6, 71.8]** |
| | | Rec | **[71.8, 15.0]** | **[80.1, 33.9, 11.0, 2.1]** | **[81.8, 61.5, 51.2, 48.8, 52.0, 67.2]** |
| | Overall Basic | Acc | **55.6** | **64.7** | **65.1** |
| | | Pre | **71.9** | **74.2** | **77.8** |
| | | Rec | **75.2** | **73.4** | **67.6** |
| | Overall Advanced | Acc | **19.3** | **17.7** | **52.1** |
| | | Pre | 28.3 | 25.4 | **64.8** |
| | | Rec | **40.2** | **29.1** | **59.9** |

achieved much lower scores than the other TMWF without time sequence input. This indicates that while our method can improve the performance of the model on samples with the same number of pages by adding time features, it can cause a sharp drop in the model's generalization ability on different page numbers.

## H SUPPLEMENTARY OPEN-WORLD EXPERIMENT

Due to the unique "first-party domain traffic isolation" policy of the Tor browser, which leads to distinct traffic patterns compared to the Chrome browser (as explained in Section 5.4), we conducted a supplementary open-world experiment on the $D_{TBB}$ dataset similar to the approach described in Section 6.4. During data collection, we observed that TBB can access around 8000 non-monitored websites (in the range of ranked websites from 100 to 10000). These websites were used as the non-monitored set for the dataset. Compared to Chrome, we have increased the upper limit of the delay range for collecting multi-tab traces from 6s to 9s due to the change in the loading time of all TBB single-tab trace samples. It's important to note that due to hardware resource limitations on the VPS, we were unable to support simultaneous access to five or more pages with multiple tabs. Therefore, the maximum number of pages in the real multi-tab traces we collected was limited to 4. Apart from these modifications, the rest of the dataset settings were kept consistent with the $D_{CHR}$ dataset. We employed the $D_{TBB-6tabs-ratio-train-40k}$ as the training set and performed predictions on various mixed multi-tab trace validation sets and real trace test sets. The suffix "mixed" in the dataset indicates combinations of multi-tab trace datasets with the same size (1000) and different page numbers (2, 3, 4). The results of this experiment are presented in Table 9.

We have observed a decline in model performance compared to the results presented in Figure 5. Additionally, we have noted a consistent trend where the model's performance on the multi-tab traces validation set ($D_{TBB-mixed-delay-val-3k}$) based on $M_{delay}$ synthesis closely resembles its performance on the real multi-tab traces test set ($D_{TBB-mixed-real-3k}$). We speculate that multiple factors contribute to the enhanced anti-WF resistance of the Tor browser. These factors may include but are not limited to, the Tor browser's first-party domain traffic isolation policy, which routes traffic from different domains through different circuits when loading tabs from distinct domains. This policy increases the variability of Tor WF to some extent, raising the learning cost for the model and demanding attackers to gather more data. Consequently, even when using datasets of equal sizes, classifiers aimed at WF for the Tor browser may exhibit weaker WF coverage compared to classifiers for other general browsers (which lack the mentioned policy), resulting in further degradation of model performance.

In other words, one of the distinctive characteristics of deep learning models, as opposed to traditional machine learning models, is their need for a large volume of samples for training. The potential reason for the observed performance degradation lies in the insufficient coverage of diverse network scenarios in the current training samples, especially given the significant impact of network conditions on WF model performance [7, 10, 20, 24] . Additionally, the first-party domain traffic isolation policy exacerbates this impact. Furthermore, considering that the Tor browser is a hardened fork of Firefox, it's possible that Firefox's traffic generation mechanisms also contribute to its resistance against WF recognition, as demonstrated in the experiments in [6] .

## I BASE RATE EXPLORATION

Based on the results presented in Appendix E and Section 6.4, we conclude that a cost-effective mode for multi-tab WF attacks involves training on multi-tab traces synthesized with $M_{ratio}$ and evaluating performance using multi-tab traces synthesized with $M_{delay}$. We continue to use the model trained on $D_{CHR-6tabs-ratio-}$

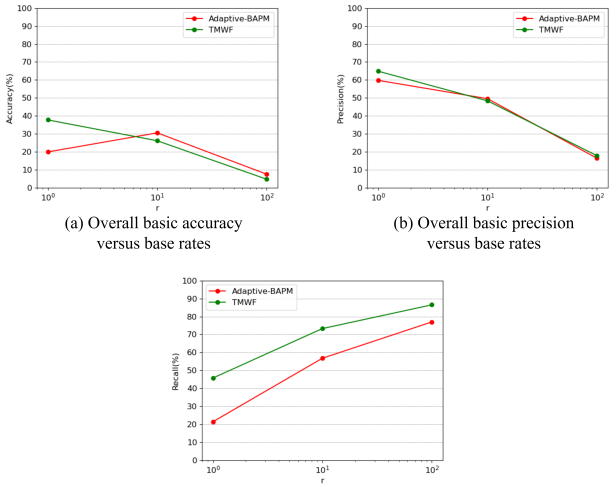
**Table 8: Performance comparison between the model with timing features added using our method and the original model.**

| Model | Metric | $D_{WT-2tabs-test}$ | | | $D_{WT-4tabs-test}$ | | | $D_{WT-6tabs-test}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Acc | Pre | Rec | Acc | Pre | Rec | Acc | Pre | Rec |
| Add Timeinfo | Overall Basic | 42.8 | 75.4 | 69.1 | 70.4 | 82.6 | 80.9 | 73.8 | 89.4 | 76.6 |
| | Overall Advanced | 15.9 | 34.5 | 37 | 21.8 | 30.8 | 35.8 | 63.7 | 84.4 | 74.3 |
| Original Model | Overall Basic | 64.2 | 80.9 | 77.4 | 74.9 | 84 | 83.9 | 75.5 | 88.9 | 78.4 |
| | Overall Advanced | 35 | 49.2 | 60.5 | 29.6 | 38.5 | 49.7 | 68.2 | 81.6 | 74.7 |

**Table 9: Classification results on multi-tab traces validation sets and real multi-tab traces test sets, based on $D_{TBB-6tabs-ratio-train-40k}$ training set.**

| Model | Metric | $D_{TBB}$ -mixed-ratio-val-3k | $D_{TBB}$ -mixed-delay-val-3k | $D_{TBB}$ -mixed-real-3k | $D_{TBB}$ -2tabs-real-1k | $D_{TBB}$ -3tabs-real-1k | $D_{TBB}$ -4tabs-real-1k |
|---|---|---|---|---|---|---|---|
| Adaptive-BAPM | Basic Accuracy | 20.9 | 10.7 | 6.7 | 4.4 | 5.1 | 9.5 |
| | Basic Precision | 68 | 57.1 | 42.1 | 19.8 | 33.6 | 46.5 |
| | Basic Recall | 22.7 | 11.8 | 7.3 | 6.0 | 5.4 | 9.2 |
| **TMWF** | Basic Accuracy | **61.3** | **31.6** | **23** | **12.7** | **16** | **34.4** |
| | Basic Precision | **75.3** | **63.8** | **58.9** | **44.3** | **54.3** | **63.2** |
| | Basic Recall | **73.5** | **31.3** | **24.8** | **15.1** | **17.3** | **35.6** |

$train - 40k$ as detailed in Section 6.4 and replace the test set with newly constructed synthetic multi-tab traces from this section. As described in Section 5.4, for each $r$ value setting encompassing multiple page numbers, we consolidate them into new datasets named $D_{CHR-mixed-delay-r1}$, $D_{CHR-mixed-delay-r10}$, and $D_{CHR-mixed-delay-r100}$. The corresponding evaluation results are presented in Figure 10.




(a) Overall basic accuracy versus base rates

(b) Overall basic precision versus base rates

(c) Overall basic recall versus base rates

**Figure 10: Model evaluation results with variations in base rate, where $r$ denotes the ratio of non-monitored traces to monitored traces.**

From Figure 10, it is evident that as the ratio ($r$) of non-monitored traces to monitored traces increases, the model's accuracy and precision show significant declines, while recall experiences a considerable improvement. We posit that this phenomenon can be attributed to the model continuing to predict input samples based on the r-value it was trained with. Consequently, false positive results increase with rising $r$ values, ultimately overshadowing true positive results. Even though TMWF achieves a recall of 86.5% at $r = 100$, its accuracy is merely 4.8%. This implies that the model's positive predictions are accurate only about once in every 20 instances. For an attacker aiming to identify user identities, the model becomes impractical as it would incur high collateral losses by acting upon its positive predictions.

Furthermore, in Figures 10-(a) and (b), we observe a rare overlap in the graphs of the two types of evaluation scores. We attribute this phenomenon to the fact that TMWF fits the proportion settings of the training data more than BAPM does. In conclusion, while TMWF does impact user anonymity under the condition of equal quantities of monitored and non-monitored traces, considering various base rates of monitored websites, achieving the recognition of these monitored websites in a multi-classification scenario remains challenging.