RAP: A Lightweight Application Layer Defense Against Website Fingerprinting

Yan Zhang, Li Yang(✉), Junbo Jia, Shirui Ying, and Yasheng Zhou

Xidian University, Xi'an 710071, China
{zhangyan_wecs,jbjia,srying,yszhou}@stu.xidian.edu.cn,
yangli@xidian.edu.cn

Abstract. Website fingerprinting (WFP) attacks threaten user privacy on anonymity networks because they can be used by network surveillants to identify webpages that are visited by users based on extracted features from the network traffic. There are currently defenses to reduce the threat of WFP, but these defense measures have some defects; some defenses are too expensive to deploy, and some have been defeated by stronger WFP attack methods. In this work, we propose a lightweight application layer defense method, *RAP*, which can resist current WFP attacks with very low data and latency overheads; more importantly, it is easy to deploy. We randomly deploy important resource files, such as JS and CSS, to multiple Tor OR servers in advance and update them regularly. By randomly scrambling the resource request order, a single request is sent and received through multiple independent paths with different Tor entry ORs. To randomize the traffic distribution, users randomly obtain the website resource files directly from the Tor node server, rather than from the original server, when browsing the website. In this way, the best attack accuracy is reduced from 98% to 53%. Additionally, to confuse the traffic, we request a small amount of additional HTML text instead of the whole website resources, which reduces the effect of state-of-the-art WFP attacks to 40% with 13% data overhead and 31% latency overhead.

Keywords: Traffic analysis · Website fingerprinting · Web privacy

1 Introduction

To conceal a user's location and usage behavior from anyone who conducts network surveillance or traffic analysis, Tor [18] provides an anonymous communication strategy through a global voluntary network that contains more than 7,000 relays. A Tor client uses an onion proxy (OP) to selects 3 onion relays (OR) in order to create a virtual encrypted data transmission tunnel called *circuit*. The three relays in the chain are called *entry*, *middle* and *exit*, respectively. Each OR

Supported by the National Natural Science Foundation of China (62072359, 62072352), the National Key Research and Development Project (2017YFB0801805).

W. Shi et al. (Eds.): SPNCE 2021, LNICST 423, pp. 254–270, 2022.
https://doi.org/10.1007/978-3-030-96791-8_19

only knows its predecessor and successor. The client separately negotiates the encryption key for communication with each OR in the chain. With the help of a key, the user's data are wrapped and encrypted like an onion, forming fixed-size chunks called *cells*.

In recent years, multiple studies [1,5,10,14–16] have shown that Tor is vulnerable to WFP attacks. A WFP attack is a special kind of traffic analysis attack that aims to identify the web content of anonymous data flow by observing traffic patterns, such as the number, direction, ordering, and timing of packets. With the development of deep learning [17,25] and computer technologytechnology [24,26], the threat of website fingerprint attacks is serious. To protect user privacy, a series of studies on defense [2–4,6,9,12] against WFP attacks have appeared. However, some defense systems add a large number of dummy packets and delay real packets, resulting in excessive latency and data overheads. Some defense systems are compromised by more advanced and effective attacks, and others cannot be deployed in the real Tor network because of the high resource costs. To counter more powerful WFP attacks, it is increasingly urgent to invent a defense.

In this paper, we propose RAP, an application layer defense that can defeat state-of-the-art WFP attacks, which not only has lower data and latency overheads, but is also easy to deploy. We find that the previous WFP attacks assume that users visit the website in the same scenario. The traffic is collected under this assumption, making the classification result very high. However, if we gather the traffic of users in different locations around the world, the accuracy will be greatly reduced. This phenomenon may be because users are located in different regions and website servers use content delivery network (CDN) technology, which leads to different load traces of the same website in different regions. Our defense uses this idea for reference and randomly distributes the static resources of the website to different Tor ORs.

Our work makes the following contributions:

(1) We design a lightweight WFP defense method, RAP, which is based on randomizing the request order and location of website resources. First, we put some unchangeable embedded resources on Tor OR nodes in advance and update them regularly so that a large number of Tor OR nodes can become our CDN servers. When a user visits a website, the HTTP request sequence of all website resources is scrambled, and the request location and sequences of website resources are scheduled. Finally, requests are sent through multiple Tor paths with different entry nodes; when the user visits the same website, the received traces are different.
(2) We explore several traffic randomization strategies that can serve as candidates for adoption in our defense. Based on four state-of-the-art WFP classifiers, we conduct real-world evaluations to analyze the efficiency of these strategies against modern WFP attacks.
(3) We conduct an extensive analysis to prove the effectiveness of our RAP defense. Our defense strategy reduces the accuracy of attacks on state-of-the-art WFP from 98% to 40%, with 13% data overhead and 31% latency overhead.

2 Related Work

2.1 WFP Attacks

In 2013, Wang and Goldberg [20] proposed replacing previous packet with Tor cells to extract fingerprints from websites. They increased the attack accuracy to more than 90%. Further works [1,5,10,14–16,19] have been proposed since then that have pushed a higher accuracy and a lower false positive rate.

In 2014, Wang et al. [19] proposed a WFP attack based on k-Nearest Neighbors (k-NN), extracting more than 3000 traffic characteristics. k-NN outperforms existing methods and reached a 91% accuracy in the closed world scenario. In 2016, Hayes and Danezis [10] proposed the k-FP classifier based on random decision forests. The K-FP classifier can correctly determine which of the 30 monitored hidden services a client is visiting with an 85% true positive rate and a false positive rate as low as 0.02%. In the same year, Panchenko et al. [14] proposed a classifier, CUMUL, which outperforms existing methods in terms of both the recognition rate and computational complexity.

Recently, several WFP classifiers based on deep learning have shown a better performance than traditional machine learning. Sirinam [16] proposed a WFP classifier, DF, with a deep convolutional neural network (CNN). DF reached an accuracy greater than 98% without defense, and it also has a very good effect on lightweight defenses, such as WTF-PAD [12] and Walkie-Talkie [22].In addition, other works [7,21–23] implement WFP attacks on multitab pages in more practical scenarios because Juarez et al. [11] criticized the assumption that the previous work is unrealistic.

2.2 WFP Defenses

To change the traffic pattern, several defenses [3,4,13,19,22] measures make use of the attacker's inability to distinguish between real and dummy data packets by adding dummy packets and delaying the real data packets to achieve the purpose of defense. Dyer et al. [8] designed Buffered Fixed Length Obfuscation (BuFLO) to achieve traffic obfuscation by sending data at a constant rate in both the sending and receiving directions. However, the huge data overhead is a flaw of this method. In 2016, Juarez et al. [12] proposed WTF-PAD, which is an implementation of adaptive padding. This defense uses a token system to generate dummy packets and fill them to the gaps of rear traces. Recently, Gong and Wang [9] proposed two zero-delay lightweight defenses, Front and Glue. Front uses WF attacks to rely on the feature-rich trace front, adding dummy packets to the trace front, while Glue uses the attacker's inability to distinguish two consecutive website split points to achieve a better effect than heavyweight defenses, such as Tamaraw. Cadena and Mitseva et al. [2] used the concept of traffic splitting to establish multiple Tor paths, send and receive packets through multiple different paths, and merge and split traffic on middle ORs.

The general defense at the application layer does not rely on adding dummy packets or delaying real packets actively. Cherubin et al. [6] proposed confusing

website traffic by adding web objects and inserting additional dummy requests, but this defense only applies to onion services. Cadena et al. [2] proposed that by establishing multiple Tor paths and using the HTTP range option, the bytes of web objects are randomly requested on each path so that an attacker who controls a malicious entry node can only observe a small amount of website traffic each time. In part, the detection rate of the WFP classifier has been reduced by nearly 50%. However, this defense provides protection against malicious entry ORs only.

3 Threat Model

Based on previous WFP research work, we take a special scenario into consideration: users use network proxies. Under these circumstances, the adversary can passively collect traffic between the user and the Tor entry OR node without modifying, delaying, or intercepting the original traffic. The attacker can always observe the complete traffic generated by tor users' surfing period, regardless of what defense the Tor user uses. The attacker knows the user's identity, but does not know which website the user is visiting. We assume that the adversary can collect traffic in the user's network agent (see Fig. 1) and can deploy a local network environment similar to the user's environment. WFP attacks usually correspond to supervised machine learning classification problems. After defining a set of websites to be detected, the adversary collects the traffic traces loaded on each website in the same scenario and extracts the fingerprint features. Each webpage is a class, and a particular trace belonging to this class is called an *instance*. Adversaries can use machine learning or deep learning to train classifiers to identify website categories. Finally, the adversary uses the same method to extract the features generated by the user's visit behavior. Then, the classifier is used to identify the website corresponding to this traffic.

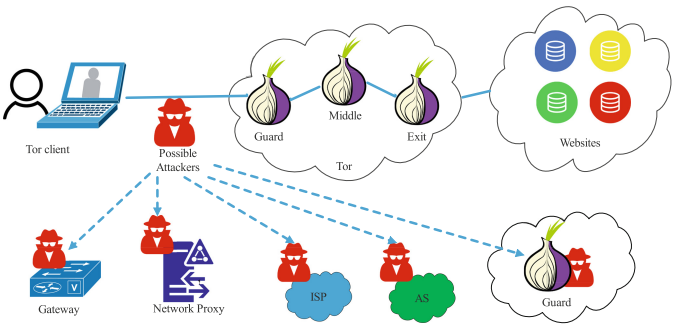

Fig. 1. The WFP threat model.

The adversary uses the same processing method to process the traffic generated by the user's visit to the website, and then uses the classifier to identify the website corresponding to this traffic. The analysis of WFP attacks generally has

two scenarios: *closed world* or the *open world* [5,10,14–16,20]. In the close world scenario, we assume that the user only visits a website in a fixed website collection, and these fixed websites become monitored websites. Here, the adversary trains on a number of traces from these sites and aims to classify different traces from the training set into one of the monitored websites. Although the closed-world is less realistic-it assumes the adversary knows every site a user visits-it is a useful measure of a classifier's ability to distinguish between websites. In a more realistic open world scenario, a user can visit any website, including websites that the adversary does not know. These sites that the adversary does not know are called unmonitored sites. The attacker only has the fingerprints of the websites he is interested in, so that the adversary is in When carrying out a WFP attack, it is not only necessary to distinguish whether the website of the unknown traffic belongs to the collection of websites of interest, but also to distinguish which website is when the unknown traffic belongs to the collection of websites of interest. In addition, the adversary can bias her classifier by training on some number of unmonitored sites. Though there is no overlap between unmonitored training and testing sites, learning how to distinguish one set of unmonitored sites often helps with others. As in the previous work, we set the adversary's capabilities: (1) The adversary knows the start and end time of the victim's visit to the website, that is, the adversary's collected traffic does not include other website traffic; (2) the adversary's collected traffic sequence There is no mixed background traffic; (3) The adversary knows that the victim uses the defense system and understands its defense strategy.

4 Defense System Design

4.1 Overview

When a user requests a website using a browser, a simple process is that the browser initiates a TCP connection with the target IP; The browser parses the HTML code and makes requests to the resource link embedded in the HTML code (such as JS, CSS, pictures, etc.), then the browser presents the page to the user. Static resources occupy a considerable part of the web page. Many websites put static resources on the CDN to facilitate user access. In fact, the network traces of accessing the CDN website in different regions may be different, which will cause the accuracy of the WFP classifier to seriously decline. Our defense makes Tor users visit each website, similar to visiting a CDN website, adding other random factors to the HTTP request, and ensuring that the load tracking of the same website is different each time.

4.2 Architecture

Our defense system is composed of volunteer OR nodes, Tor directory servers (DS), and user agents. After we have determined the collection of websites to be

defended by WFP, the Tor directory server will visit the websites in the collection one-by-one, recording the JS resource objects, CSS resource objects, and logo image objects. Because these resource objects are not easy to change. The Tor directory server reads the list of Tor ORs that are willing to provide defenses and builds a table of correspondence between the website resource objects and ORs (see Algorithm 1). The web object is renamed to a unique name, and then, we associate the object with multiple OR nodes to prevent some OR nodes from going offline. The directory server regularly revisits the website, updating the corresponding relationships between the website resources and OR nodes. The corresponding relationship table is called the *corTable* and is stored locally before synchronizing to other directory servers. The HTTP service is started in the selected Tor node, and the Tor node can be deemed the HTTP server. It regularly obtains the *corTable* from the directory server and finds resource objects related to itself, according to the *corTable*. The server downloads the web resource object using the resource links embedded in the original HTML code and renames the files according to the table information. Finally, resource objects are saved in the server. To reduce the storage burden, once the *corTable* is obtained from the directory server and the resource objects are downloaded the old version resource objects stored in the server can be deleted.

Algorithm 1. Generate the corresponding relationship table of web resources and OR nodes.

Input: Collection of websites that need to be defended, $webSet$;
Output: Corresponding relationship table of web resources and OR nodes, $corTable$;
1: Initialize global $corTable$;
2: **for** each $w \in webSet$ **do**
3: DS request w to get $HTML$;
4: $jcSet$ = Parse($HTML$);
5: **for** each $f \in jcSet$ **do**
6: Random select ORs from $ORSet$;
7: $newName$ = UUID + (.js|.css);
8: I_w = combination($w, f, newName, ORs$);
9: $corTable$.add(I_w);
10: **end for**
11: **end for**
12: **return** $corTable$;

When a user visits a website, the user usually sends a series of HTTP requests in sequence through a Tor circuit to obtain all the web objects needed to display the site. Under our defense, we first start multiple OP instances, each of which maintains a three-hop circuit, which is built using the existing Tor circuit creation concept. When a user visits a website, the original website HTML text is obtained through an HTTP request. After the HTTP request is parsed, the web object HTTP requests are not sent to the original resource server; instead, requests are sent to the RAP-local defense system (the defense system can be

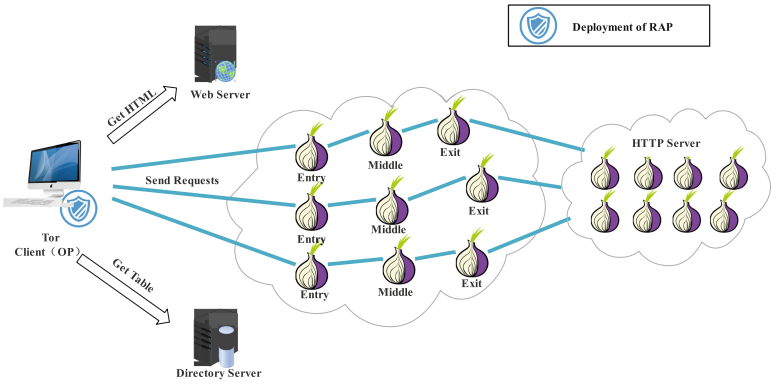

Fig. 2. Clients request websites from different servers through multiple paths.

deemed as a local proxy plug-in between the user's browser and the user's OP). As shown in Fig. 2), our defense operations are as follows:

- It completely shuffles all web HTTP requests, including requests for web embedded objects, such as JS, CSS, etc.
- It finds the relationships between resource objects and ORs according to the *corTable*. The OR node is randomly selected, which has the requested resource objects, and then a new HTTP request is constructed.
- It sends the web HTTP requests through multiple three-hop circuits established in advance according to our strategy. These requests are sent in parallel, and the web server responses are received through the request circuit.

4.3 Traffic Randomization Strategy

The main challenge of our defense strategy is to make the load traces of the same page highly different to prevent the adversary from finding fingerprint patterns. We analyze the influencing factors from three aspects: server, transmission path and client.

(1) After we determine the size of the website collection, we can obtain the website source code and obtain the number and size of all the resource files that need to be stored on the Tor OR nodes. What we need to care about is the proportion of each website's resource object to be stored on the Tor nodes, as well as how many Tor relay nodes need to be used to achieve a balance between the performance overhead and defense effects. From our preliminary cognition, using more Tor nodes can achieve better results. Therefore, we choose Tor OR nodes that are willing to provide services for us and discuss the proportion of website resource objects on Tor nodes in subsequent experiments. This means that the web resource object scheduler on the directory server selects a certain number of embedded resources to Update, according

to the given proportion. Defining the size of JS or CSS on the website as s_r, the number of stable and unchangeable resources that the website can put on the Tor OR as n_{st}, and the number of resources that are actually put on the Tor OR as n_t, we analyze the proportion P that needs to be scheduled.

$$P = \frac{\sum_{j=1}^{n_t} s_{ri}}{\sum_{i=1}^{n_{st}} s_{ri}} \tag{1}$$

(2) We analyze the traffic randomness of the number of parallel three-hop circuits used to send requests and receive responses. Although using a large number of Tor circuits with different entry ORs can reduce the amount of information available to each entry OR node and sending requests in parallel can reduce the time to receive the last HTTP request, our defense purpose is not to reduce the sequence of traffic passing on the single entry OR node. A large number of three-hop circuits will increase the local load of the client, so we set the initial number of Tor circuits to 5, according to Caneda's [2] suggestion, and then increase or decrease the number in turn to analyze the effect.

(3) We analyze the impact of different HTTP request allocation strategies on multiple circuits on the traffic randomness. Our basic scheme is round robin. We further consider a batched weighted random strategy. For all HTTP requests of a web page, we create an n-dimensional vector that is composed of the maximum number of requests sent by each circuit. We choose the probability density function of the *exponential distribution* to obtain five random numbers, and then we obtain n probabilities by weighting, the sum of which is 1. Finally, we multiply the probability according to the number of HTTP requests to obtain the random number of n-dimensional vectors and then randomly select a Tor circuit for each HTTP request. When the number of requests sent by a Tor circuit reaches the threshold set in advance, we will not use this circuit to send requests. Each time we visit the website, we will choose a new random number seed for the exponential distribution to ensure the unpredictability of the tracking of each page's loading process.

5 Experimental Setup

5.1 Dataset Description

To evaluate our WFP defense system, we deployed our defense system in the real Tor network environment. We collected a new dataset between September and November 2020 using Tor Browser 10.0 with a configured SSR proxy. A *Python* script is used to drive the Tor browser to automatically visit the website with the help of the *tcpdump* tool to record the original traffic and extract the data at the Tor cell level. The traces we collected and used for experiments were not generated by real users, so there is no moral concern. We selected 20 Tor ORs as our volunteer web resource object server, using another Tor OR node with a resource scheduling system as our directory server.

For the closed world scenario, we chose Alexa's top 100 websites to form our website collection. We first collected the dataset without applying our defenses and collected 100 rounds of the homepage of each website for 100 websites, which is called *DS-NODEF*. We deleted the trace that failed to load the page due to client or server errors, and we adopted the same solution for other subsequent datasets. For our RAP defense, we selected our traffic-randomization strategy to collect 100 traces for each website. We call this dataset *DS-DEF*. Finally, we visited each of Alexa's 10000 most popular websites once, excluding the first 100 sites used to build our closed world dataset, *DS-NODEF-BG*, without applying our defenses, and we used these websites as background for our open world evaluation. Note that all the traffic we collect has passed through the SSR network proxy. SSR can encrypt, obfuscate, and forward the original traffic. We call the dataset we collected on the Tor browser the SSR proxy *SSR-TOR-DS*. The dataset collected by Rimmer [15] is named *Rimmer-DS*, which is a comparison dataset.

5.2 Classifiers and Evaluation Setup

For our evaluation, we considered four state-of-the-art WFP attacks: k-NN [19], CUMUL [14], k-FP [10], and DF [16]. k-NN, CUMUL, and k-FP manually extract the representative features and then use machine learning methods for classification. DF directly uses the sending and receiving sequence of cells to input into the deep convolutional neural network for classification. In addition to the suggested parameters given in the original text, we also refer to the parameters given in this work [9]. We apply 10-fold cross-validation with respect to the total number of collected page loads.

We assume that the attacker can collect traffic at the user's network proxy, knows our defense strategy, and has sufficient resources to collect data and train classifiers. The higher the value of FPR is, the safer our defense system is. For our closed world analysis, we computed the accuracy, i.e., the probability of a correct prediction (either true positive or true negative). We calculated the TPR, i.e., the fraction of accesses to foreground pages that were detected, and the FPR, i.e., the probability of false alarms, for our open world experiments.

6 Evaluation and Discussion

6.1 Analysis of RAP Defense

Influence of Shuffling HTTP Requests and Multipath. First, to analyze the effect in the undefended scenario, we conducted attack experiments in the DS-NODEF dataset, adding proxy obfuscation encryption in comparison with Rimmer's dataset. With the results shown in Table 1, we found that after adding proxy obfuscation encryption, the accuracy of the four methods remained stable and did not decrease significantly; they even have a similar effect to the DF attack. This might be because the CNN model used by DF has the ability to

learn the pattern of proxies, which shows that the SSR proxy cannot provide an effective defense. We compare the experimental results using our datasets and other datasets, which do not use the SSR proxy.

Table 1. Accuracy (in %) of state-of-the-art WFP attacks in scenarios without defense and against our simple traffic randomization strategies.

	Rimmer-DS undefended	SSR-TOR-DS Undefended	Multipath	Shuffle HTTP requests and multipath
k-NN	98.2	90.02	88.47	27.2
CUMUL	98.5	92.1	91.23	59.2
k-FP	98.4	94.42	90.86	71.36
DF	98.75	98.11	92.32	73.76

As shown in Table 1, we performed simple multipath defense experiments in the SSR-TOR-DS dataset, which means that we send the complete HTTP requests via five circuits based on the round robin method. Under this situation, we found that the degree of decrease for the four methods is relatively low, which verifies the conclusion of a separate study [2]. Then, we performed another experiment, which sent the HTTP requests after shuffling and analyzed the influence. The results illustrate that this strategy has more influence on the accuracy of the k-NN classifier; it decreases to 27.2%, and the accuracy of other classifiers decreases by 20%–30%, to some extent.

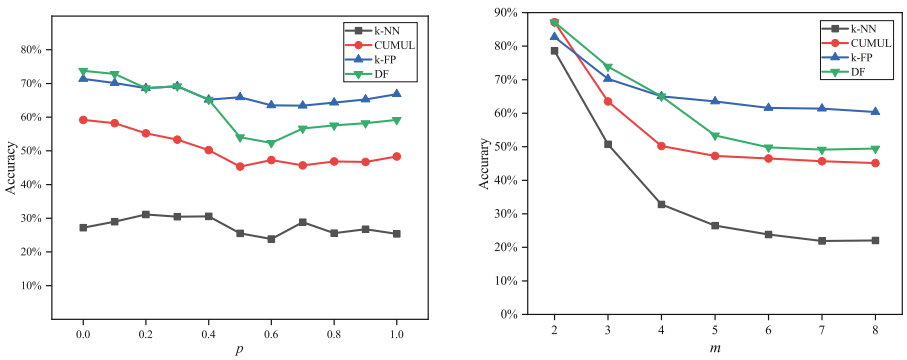

Fig. 3. Accuracy of state-of-the-art WFP attacks under different resource ratios p and circuit numbers m.

Efficiency of Resources Proportion on Tor Servers. We perform some experiments to determine the influence of resource proportion on the DS-DEF dataset. When $P = 0$, the order of HTTP requests is disrupted, and the requests are sent to the original website through multiple circuits using the round robin method. The result is shown in Fig. 3(a). We find that web object requests from different servers have little influence on the accuracy of k-NN attacks. This may be because the accuracy of k-NN attacks dropped sharply before requesting data from the Tor HTTP server. However, the accuracy of the other three attack methods has decreased significantly. When P exceeds 0.6, the drop rate decreases, or even no longer decreases, and the DF attack appears to rise. Although it cannot be stated directly that $P = 0.6$ is the optimal choice, we can almost achieve the best results when $P = 0.6$. Therefore, we set P to 0.6 in subsequent experiments.

Efficiency of the Number of Multipaths on Traffic Randomization. To analyze the impact of m (the number of multipaths) in the defense system, we set P (the resource ratio of the Tor HTTP server) to 0.6. The client's requests were randomly shuffled and then sent using the round robin method. Based on the recommendation of TrafficSliver, first the number of circuits is set to 5. To find its best range, we perform tuning on the basis of 5. Finally, we set the number of circuits $m \in [2, 8]$ to test our classification accuracy. As shown in Fig. 3(b), we found that when m is larger, the classification accuracy decrease rate is larger, and when m is greater than 5, the decrease rate of the classifier decreases. Therefore, we conclude that $m = 5$ is a good choice because 5 Tor paths will not significantly increase the circuit setup time; in addition, the defense effect is still acceptable.

Efficiency of Batched Weighted Random. We chose $P = 0.6$, and the client requests were randomly scrambled and sent through 5 circuits. We tested the impact of sending requests through round robin and batched weighted randomization on traffic randomization. Due to the small number of HTTP requests on some websites, to facilitate comparative experiments, we removed websites with fewer than 20 HTTP requests from Alexa's top 100 website list and selected and added 100 from the subsequent Alexa's top websites. Then, each website was collected 100 times and the dataset was reconstructed. Our experimental results are shown in Table 2. We found that batch weighted randomization has better defense effects than round robin, and that the accuracy of the four classifiers decreases by an average of 10%, which means that our batch randomization requests are effective, but the accuracy of k-FP attacks is still above 50%. This may be because RAP defense has not greatly changed the overall size and time of website traffic, resulting in k-FP finding the overall relationship of traffic packets.

Efficiency of Adding Additional Requests. In the initial stage of the traffic generating process, RAP sends HTML request to original website and *corTable*

Table 2. Accuracy (in %) of state-of-the-art WFP attacks in scenarios without defense and against our defense strategies.

	k-NN	CUMUL	k-FP	DF
Undefended	90.02	92.10	94.42	98.11
Round Robin	23.84	47.28	63.52	52.37
Batched Weighted Random	**14.25**	**38.38**	**53.54**	**42.80**

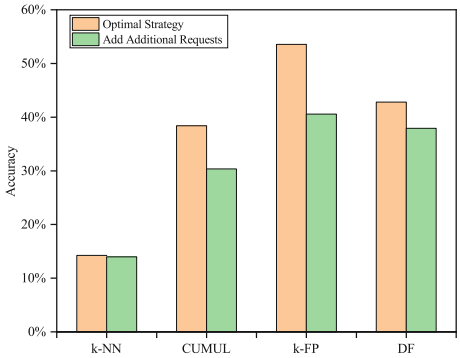

Fig. 4. Accuracy (in %) of state-of-the-art WFP attacks in different scenarios of optimal strategy and additional requests.

requests to. We think that this period is helpful for extracting the pattern of WFP attacks, so we decide to confuse the traffic at this stage. Our idea is to send a random number of HTML requests to the randomly selected websites in Alexa's top 10000 at the same time as sending original HTML requests. We set the number between 1 and 5. The results are shown in Fig. 4. We further reduce the accuracy of WFP attacks by increasing head obfuscation. Although the reduction effect is not good for k-NN and DF, the accuracy of k-FP attacks can be reduced to 40.58%. Our method is a successful defense for attackers who have sufficient traffic data. In conclusion, it is effective to add obfuscation in requests to reinforce our defense system.

Open World Evaluation. We use our optimal strategy [$P = 0.6$, $m = 5$, batch randomized requests, obfuscated headers] to evaluate the defense system in open world scenarios. As a baseline, we first used the non-defended dataset DS-NODEF as a foreground set and DS-NODEF-BG as our background set and computed TPR and FPR for each classifier. As shown in Table 3, the overall effect of our defense system in the open world scene can reach an advanced level, and our defense is effective against k-NN and CUMUL classifiers. In the face of more powerful k-FP and DF attacks, our FPR is only 61.52% and 53.42%, respectively. This may be because k-FP uses many time features, and the deep neural network structure of DF can find the defects of our defense.

Table 3. TPR and FPR (in %)of state-of-the-art WFP attacks against our defense in the open world.

	k-NN		CUMUL		k-FP		DF	
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
Undefended	88.21	15.84	93.85	12.30	92.56	6.50	96.70	5.23
RAP	20.32	76.89	42.36	74.36	53.09	61.52	63.71	53.42

6.2 Comparison to Prior Defenses and Overhead

Finally, we compare other popular WFP defenses, such as Tamaraw [4], WTF-PAD [12] and TrafficSliver-App [2], with our defense system with the best strategy. It should be noted that even if our dataset has more encryption and confusion of SSR network agents compared to other defense datasets, the influence of SSR is small, and it can only reduce the accuracy of some classifiers by less than 10%. We directly use the attack effect data in the previous defense work [2].

Overhead Measurement Method. Defining the number of packets as $|P|$, the timestamp of the i-th packets as t_i, the direction and size of the i-th packets as L_i. The packet sequence generated during website loading can be expressed as:

$$P = <(t_1, L_1), (t_2, L_2), \ldots, (t_i, L_i), \ldots, (t_{|P|}, L_{|P|})> \tag{2}$$

The latency overhead $T(D)$ generated by defense system D on packet sequence P refers to the additional time generated by transmitting real packets divided by the original total transmission time. It is assumed that the time of the last real packet in defense traffic P' is t_k. Then $T(D)$ is defined as follows:

$$T(D) = \frac{t_k - t_{|P|}}{t_{|P|}} \tag{3}$$

The data overhead $O(D)$ refers to the total amount of virtual packets divided by the total amount of real data. $O(D)$ is defined as follows:

$$O(D) = \frac{\sum_{i=1}^{|P'|} |L_i| - \sum_{i=1}^{|P|} |L_i|}{\sum_{i=1}^{|P|} |L_i|} \tag{4}$$

Security Against State-of-the-Art WFP Attacks. Table 4 shows the attack effect of state-of-the-art WFP attacks in the closed world scenario. Our RAP defense effect is obviously better than that of WTF-PAD, and WTF-PAD was once considered the optimal defense strategy. Compared with Tamaraw, the effect of our defense is significantly weakened. However, the defense effect of Tamaraw depends on the data and latency overhead. Our defense system is

not as good as TrafficSliver-APP in counter with CUMUL and k-FP classifiers, but we believe that the effect can meet the requirements of the Tor defense system, and the accuracy of CUMUL and kFP classifier can be reduced to 30.35% and 40.58%, respectively. It is worth noting that the accuracy of the DF classifier can be reduced by our RAP defense to 37.91%, which is better than that of TrafficSliver-App. Therefore, we conclude that our RAP defense can provide excellent defense effects.

Performance Overhead. In addition to measuring the defense effect, performance overhead is also crucial to the WFP defense system. If the website takes too much time to load, users may have bad experiences. Therefore, we measure the data and latency overhead generated by the defense, comparing it with the previous WFP defense. As shown in Table 5, compared with all previous WFP defenses, our RAP defense only has a small data overhead. This part of the overhead is generated by HTTP requests added when requesting the relationship table from the directory server, as well as obtaining the initial web HTML. In terms of latency, our defense also has a good performance, which is similar to TrafficSliver, because we do not artificially add a delay to HTTP requests. The delay comes from sending HTTP requests to our defense proxy, which influences the order of HTTP requests and selects circuits. In addition, our defense overhead is much lower than Tamaraw. Although WTF-PAD does not add any time delay, it does not provide sufficient security to defend against WFP attacks. In conclusion, our defense system only adds a small amount of overhead, and

Table 4. Accuracy (in %) of state-of-the-art WFP attacks against our RAP defense and other prior defenses.

	k-NN	CUMUL	k-FP	DF
Undefended	90.02	92.10	94.42	98.11
Tamarraw	4.86	6.86	5.50	4.11
WTF-PAD	35.23	75.73	67.50	85.62
TrafficSliver-App	14.93	24.13	28.72	57.34
RAP	13.98	30.35	40.58	37.91

Table 5. Latency and data overhead (in %) created by WFP defenses.

	Latency Overhead	Data Overhead
Undefended	0	0
Tamaraw	78.43	162.93
WTF-PAD	0	32.71
TrafficSliver-App	31.23	0
RAP(optimal)	31.75	13.60

these overheads are reasonable. We analyze the overhead of randomized HTTP requests to schedule a web object in the Tor network. We select the dataset with a web object ratio of 0.6 for analysis. Because there are very few logo images and other files that are not easy to change, we only analyze JS, CSS and file scheduling overhead. Our 100 websites have 1396 JS and CSS files, occupying 161 MB of space. On average, there are 13.96 resource files per website, 0.1153 MB per resource file, and 1.61 MB per website resource. We select P as 0.6 and back up 5 copies for each resource; that is, we select 1396*0.6*5 = 4188 files to put on 20 Tor or HTTP servers. Each relay node needs 209.4 MB of space to store our selected web resource files. If we place these resource files on 2000 Tor nodes, each Tor node only needs 2.094 MB of additional storage space. In fact, we need to use more than 200 MB of storage space to install Tor service on the host, so the space of these web resources is reasonable. As the web resource files may change when the website content is updated, the resource scheduler that we use to collect our dataset will update all the website resource files once an hour, which will have a bandwidth of 4188*0.1153/60/60 = 0.1341 MB/s. According to our observation, except for a small number of websites that often change the embedded resources, the change frequency of JS or CSS files for most websites is less than once a day; therefore, the Tor bandwidth cost of RAP may be far less than our estimation.

6.3 Discussion and Limitations

Although our defense system achieves a good defense performance with very low latency and bandwidth overhead, we may encounter the following problems: the owner of the Tor node may damage the resources maliciously or modify the resource objects. We suggest that each Tor server that stores web objects calculates the checksum for resource files, letting the client request resource files from one Tor server and obtain the checksum from another Tor server to verify the integrity of the web resource objects. One thing that may be criticized about our defense system is the inability to deploy defense to all site collections because there are a limited number of Tor nodes with an unlimited number of resource files. In fact, the arms race for fingerprint attacks and defenses on websites has never stopped. Many useful defenses in the past, such as WFP-PAD, cannot resist DF attacks anymore, and our defense can resist only existing methods of fingerprint attacks on a limited collection of websites.

7 Conclusion

We proposed a lightweight application-level RAP defense to protect against WFP performed by possible adversaries. RAP is based on the idea of CDN, where web objects are placed on the Tor node, clients obtain HTML files from the original site, all HTTP requests are obtained and sent through multiple Tor paths, some requests are sent to the TOR node instead of the original site, and additional HTTP requests are added to confuse traffic during the initial stage of

visiting the site. Because RAP is a defense means acting on the application layer, it has scalability and compatibility in theory. We analyzed the traffic randomization scheme and determined the optimal system parameters. We showed that our RAP defense is able to reduce the accuracy from more than 98% to 40% for all state-of-the-art attacks, with only 31% latency and 13% data overhead. Our defense system does not need to modify the underlying Tor network. Therefore, RAP is suitable for deployment in Tor.

References

1. Bhat, S., Lu, D., Kwon, A., Devadas, S.: Var-cnn: a data-efficient website fingerprinting attack based on deep learning. Proc. Priv. Enhanc. Technol. **4**, 292–310 (2019)
2. De la Cadena, W., et al.: Trafficsliver: fighting website fingerprinting attacks with traffic splitting. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pp. 1971–1985 (2020)
3. Cai, X., Nithyanand, R., Johnson, R.: Cs-buflo: a congestion sensitive website fingerprinting defense. In: Proceedings of the 13th Workshop on Privacy in the Electronic Society, pp. 121–130 (2014)
4. Cai, X., Nithyanand, R., Wang, T., Johnson, R., Goldberg, I.: A systematic approach to developing and evaluating website fingerprinting defenses. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 227–238 (2014)
5. Cai, X., Zhang, X.C., Joshi, B., Johnson, R.: Touching from a distance: website fingerprinting attacks and defenses. In: Proceedings of the 2012 ACM conference on Computer and communications security, pp. 605–616 (2012)
6. Cherubin, G., Hayes, J., Juarez, M.: Website fingerprinting defenses at the application layer. Proc. Priv. Enhanc. Technol. **2017**(2), 186–203 (2017)
7. Cui, W., Chen, T., Fields, C., Chen, J., Sierra, A., Chan-Tin, E.: Revisiting assumptions for website fingerprinting attacks. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, pp. 328–339 (2019)
8. Dyer, K.P., Coull, S.E., Ristenpart, T., Shrimpton, T.: Peek-a-boo, i still see you: why efficient traffic analysis countermeasures fail. In: 2012 IEEE Symposium on Security and Privacy, pp. 332–346. IEEE (2012)
9. Gong, J., Wang, T.: Zero-delay lightweight defenses against website fingerprinting. In: 29th {USENIX} Security Symposium ({USENIX} Security 20), pp. 717–734 (2020)
10. Hayes, J., Danezis, G.: k-fingerprinting: a robust scalable website fingerprinting technique. In: 25th {USENIX} Security Symposium ({USENIX} Security 16), pp. 1187–1203 (2016)
11. Juarez, M., Afroz, S., Acar, G., Diaz, C., Greenstadt, R.: A critical evaluation of website fingerprinting attacks. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 263–274 (2014)
12. Juarez, M., Imani, M., Perry, M., Diaz, C., Wright, M.: Toward an efficient website fingerprinting defense. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9878, pp. 27–46. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45744-4_2

13. Nithyanand, R., Cai, X., Johnson, R.: Glove: a bespoke website fingerprinting defense. In: Proceedings of the 13th Workshop on Privacy in the Electronic Society, pp. 131–134 (2014)
14. Panchenko, A., et al.: Website fingerprinting at internet scale. In: NDSS (2016)
15. Rimmer, V., Preuveneers, D., Juarez, M., Van Goethem, T., Joosen, W.: Automated website fingerprinting through deep learning. arXiv preprint arXiv:1708.06376 (2017)
16. Sirinam, P., Imani, M., Juarez, M., Wright, M.: Deep fingerprinting: undermining website fingerprinting defenses with deep learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1928–1943 (2018)
17. Sun, J., Wang, X., Xiong, N., Shao, J.: Learning sparse representation with variational auto-encoder for anomaly detection. IEEE Access 33353–33361 (2018)
18. Syverson, P., Dingledine, R., Mathewson, N.: Tor: the secondgeneration onion router. In: Usenix Security, pp. 303–320 (2004)
19. Wang, T., Cai, X., Nithyanand, R., Johnson, R., Goldberg, I.: Effective attacks and provable defenses for website fingerprinting. In: 23rd {USENIX} Security Symposium ({USENIX} Security 14), pp. 143–157 (2014)
20. Wang, T., Goldberg, I.: Improved website fingerprinting on tor. In: Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society, pp. 201–212 (2013)
21. Wang, T., Goldberg, I.: On realistically attacking tor with website fingerprinting. Proc. Priv. Enhanc. Technol. **4**, 21–36 (2016)
22. Wang, T., Goldberg, I.: Walkie-talkie: an efficient defense against passive website fingerprinting attacks. In: 26th {USENIX} Security Symposium ({USENIX} Security 17), pp. 1375–1390 (2017)
23. Xu, Y., Wang, T., Li, Q., Gong, Q., Chen, Y., Jiang, Y.: A multi-tab website fingerprinting attack. In: Proceedings of the 34th Annual Computer Security Applications Conference, pp. 327–341 (2018)
24. Yang, L., Li, C., Wei, T., Zhang, F., Ma, J., Xiong, N.: Vacuum: an efficient and assured deletion scheme for user sensitive data on mobile devices. IEEE Internet Things J. 1 (2021)
25. Yi, B., et al.: Deep matrix factorization with implicit feedback embedding for recommendation system. IEEE Trans. Ind. Inf. **15**(8), 4591–4601 (2019)
26. Zhang, J., Yang, L., Yu, S., Ma, J.: A dns tunneling detection method based on deep learning models to prevent data exfiltration. In: Liu, J.K., Huang, X. (eds.) NSS 2019. LNCS, vol. 11928, pp. 520–535. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36938-5_32