AAP: Defending Against Website Fingerprinting Through Burst Obfuscation

Zhenyu Yang[1], Xi Xiao[1,4](✉), Bin Zhang[2], Guangwu Hu[3], Qing Li[2], and Qixu Liu[4]

[1] Shenzhen International Graduate School, Tsinghua University, Beijing, China
yangzy20@mails.tsinghua.edu.cn, xiaox@sz.tsinghua.edu.cn
[2] Peng Cheng Laboratory, Shenzhen, China
bin.zhang@pcl.ac.cn
[3] Shenzhen Institute of Information Technology, Shenzhen, China
hugw@sziit.edu.cn
[4] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
liuqixu@iie.ac.cn

Abstract. Website fingerprinting enables eavesdroppers to identify the website a user is visiting by network surveillance, even if the traffic is protected by anonymous communication technologies such as Tor. To defend against website fingerprinting attacks, Tor provides a circuit padding framework as the official way to implement padding defenses. However, the circuit padding framework can not support additional delay, which makes most defense schemes unworkable. In this paper, we study the patterns of HTTP requests and responses generated during website loading and analyze how these high-level features correlate with the underlying features of network traffic. We find that the HTTP requests sent and responses received continuously in a short period of time, which we call HTTP burst, have a significant impact on network traffic. Then we propose a novel website fingerprinting defense algorithm, Advanced Adaptive Padding(AAP). The design principle of AAP is similar to Adaptive Padding, which works by obfuscating burst features. AAP does not delay application packets and is in line with the design philosophy of low latency networks such as Tor. Besides, AAP uses a more sensible traffic obfuscation strategy, which makes it more effective. Experiments show that AAP outperforms other zero-delay defenses with moderate bandwidth overhead.

Keywords: Website fingerprinting defense · Tor · Circuit padding framework · Traffic analysis

1 Introduction

Tor is an anonymous communication system based on the second-generation onion router [1]. In the Tor network, communication data is first encrypted at multiple layers and then forwarded by several proxies called onion routers. Onion routers are randomly selected and no single node can know the IP addresses of both the source and the destination. The purpose of Tor is to protect people from third-party trackers, surveillance, and censorship. Due to its high security, easy deployment, and low latency, Tor has become the most popular anonymous communication system today. Tor has over 6,000 intermediate server nodes worldwide, and over 3 million people use Tor clients to communicate anonymously. However, recent studies [2] have shown that Tor is not resistant to website fingerprinting attacks. Using the Tor network to access the web is still at risk of privacy leakage. Website fingerprinting attacks have been proven to be applied in real scenarios with a high success rate [3].

Most existing defenses work in the network layer by traffic padding and traffic delaying, they incur too much bandwidth overhand or latency overhead that makes them difficult, even impossible, to deploy in real-world environments. To counter website fingerprinting attacks, Tor provides a circuit padding framework for implementing defenses. The framework does not provide any mechanism to delay actual user traffic deliberately, which makes delay-based defenses unworkable.

Tor developers prefer Adaptive Padding [4] style defenses, the most famous of which is WTF-PAD [5]. However, WTF-PAD fails to defend against advanced website fingerprinting attacks. In this paper, we analyze burst features in the network traffic and propose a new website fingerprinting defense scheme *Advanced Adaptive Padding*(AAP). AAP obfuscates traffic features by sending fake bursts in the gap between real bursts, changing the size and number of bursts. It only sends dummy packets without delaying user traffic. Besides, AAP has a finite state machine like WTF-PAD, which makes it can be deployed by circuit padding framework. We conduct comprehensive experiments to evaluate AAP. Experimental results show that AAP can achieve a good defensive effect with moderate overhead.

In summary, we make the following contributions:

– We analyze the distribution patterns of HTTP requests and responses during the loading of different websites and their correlation with network traffic. We find that the HTTP requests sent and responses received continuously in a short period of time, which we call HTTP burst, have a significant impact on network traffic.
– We propose a novel website fingerprinting defense scheme called AAP. AAP changes burst patterns of website traffic by sending dummy packets without delaying user traffic. Our experiments show that AAP can outperform other zero-delay defenses with moderate bandwidth overhead.
– We collect a new dataset in the live Tor network. The dataset contains in total of 20,000 instances with 100 monitored websites (each load 100 times) and 10,000 non-monitored websites (each loaded once).

The structure of this paper is as follows. We first introduce the background knowledge in Sect. 2, then we discuss related works in Sect. 3. We illustrate the motivation in Sect. 4 and details of AAP in Sect. 5. We introduce the dataset we collected in Sect. 6. Then we present the experiment settings and results in Sects. 7 and 8. Finally, we summarize our work in Section 9.

2 Background

In this section, we report preliminary notions related to Tor and website fingerprinting largely used in the paper, with the aim to make the paper self-contained.

2.1 Tor

Tor [1] is currently the most popular anonymous communication system, which consists of more than 6,000 volunteer nodes distributed around the world. The anonymity of Tor is guaranteed by multiple proxies. The proxy in Tor is called onion router (OR). A Tor circuit contains three ORs selected from volunteer nodes randomly, each of which knows only the IP addresses of the previous and next hops. These three ORs are the entry node, middle node, and exit node. Application traffic is transmitted through three ORs in turn, so that no one can obtain the identity of end users at the same time. The structure of the Tor circuit is illustrated in Fig. 1.

Tor sends data using 512-byte fixed-length cells. Each cell contains a header and a payload. There are two fields in the header, the first field is circuit ID, indicating which circuit the cell belongs to, and the second is the command field, indicating the type of the cell. Based on their command, cells are either control cells or relay cells. The control cell is responsible for circuit establishment and destruction, and each OR needs to resolve and perform related operations. The relay cell carries end-to-end stream data.

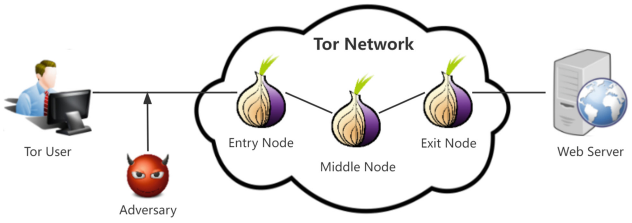

Fig. 1. Tor circuit and website fingerprinting attack model.

2.2 Website Fingerprinting

Website fingerprinting is a technique for traffic analysis. It can be seen as a classification problem that determines which website a user has visited by analyzing encrypted network traffic. Potential attackers include anyone that can sniff the communication between the user and the entry node of Tor, such as Internet Service Provider (ISP) and local network administrator because these individuals have access to the user's IP address and can observe the network traffic. Figure 1 shows the threat model of website fingerprinting. The attacker is passive, meaning that he only observes and records the traffic traces that pass through the network and does not have the ability to drop, delay, or modify packets in the traffic stream.

In order to conduct a website fingerprinting attack, the attacker needs to collect a dataset of website traffic first. Then he can use the dataset to train a machine-learning-based or deep-learning-based classifier. Finally, the attacker sniffs network traffic when victims visit websites and uses the trained classifier to determine which websites victims have visited.

3 Related Work

Website fingerprinting defenses defend against website fingerprinting attacks by changing traffic patterns observed by attackers. To ensure that website content is loaded correctly, the original application traffic cannot be modified or discarded. Common methods of website fingerprinting defense include sending dummy packets and delaying packet delivery.

Regularization defenses use fixed patterns for sending packets, they minimize information leakage and provide strong security guarantees. BuFLO family defenses [6,7] send packets with the same packet length and time interval. They send data even after the website has finished loading to mask the total load time. TAMARAW [8] performs best among them by using different sending rates at different directions. Lu et al. [9] introduce DynaFlow, which is a defense with dynamically-changing intervals and fixed burst patterns. GLOVE [10] and Supersequence [11] cluster websites according to their similarity, then calculate the super sequence of each class. The traces in the same class are morphed into the super sequence so they can not be distinguished. Walkie-Talkie [12] modifies the browser in half-duplex mode to produce easily moldable burst sequences, then it uses burst molding to change burst patterns. Regularization defenses typically have high latency and bandwidth overhead and are therefore not suitable for low overhead networks like Tor.

Distribution-based defenses work by sending dummy data to change the traffic features. Some function at the network layer and others function at the application layer. Application layer defenses include Decoy [13], HTTPOS [14], LLaMA [15] and ALPaCA [15]. While a website is loading, Decoy loads another page in the background to camouflage. HTTPOS uses a variety of strategies such as enabling HTTP pipeline and sending invalid requests to change traffic patterns. It was defeated by Cat et al. [16]. LLaMA is a client-side website fingerprinting defense, it works by adding extra delay to HTTP requests and sending redundant requests. ALPaCA is a server-side website fingerprinting defense, it works by morphing the size of HTTP objects in the web server, which makes it hard to deploy. Others work in the network layer. WTF-PAD [5] is an approach based on adaptive padding [4]. It decides whether to send dummy packets based on the difference between the sampled time interval and the real time interval. However, it fails to defend against deep learning-based website fingerprinting attacks [2]. FRONT [17] is a lightweight defense method proposed in recent years. It aims to obfuscate the feature-rich front part of the trace.

Multipath defenses establish multiple links to transmit a website through different links, ensuring that an attacker can only observe part of the traffic. Henri et al. [18] introduce a defense named HyWF that exploits multihoming. HyWF requires user devices to establish multiple different physical links to the entry node and assign network traffic to different physical links. Another multipath defense is TrafficSliver [19]. TrafficSliver requires users to establish multiple Tor connections simultaneously, each passing through a different entry node, so a malicious entry node can only get part of the traffic. Multipath defenses require extra infrastructure or modify Tor's protocol. Therefore, they are difficult to be implemented.

In recent years, deep learning has made rapid progress and achieved excellent results in the field of website fingerprinting. Many defenses use adversarial-based techniques to defend against deep learning-based website fingerprinting attacks. Hou et al. [20] propose WF-GAN to fight back against website fingerprinting attacks. The structure of WF-GAN is based on AdvGAN [21] with improvements. Mockingbird [22] fools deep learning-based classification models with adversarial examples. Both of them take the entire web traffic trace as input to produce perturbations which makes them impossible to deploy in practice. Other adversarial-based defenses [23] achieve good defensive effectiveness with a small amount of overhead. However, if the attacker knows the defense scheme used by the victim and trains attack models with the defended data, adversarial-based defenses become ineffective.

4 Motivation

There are two commonly used methods in designing website fingerprinting defense algorithms.

- Make different website traffic the same, including converting the traffic pattern from one website to another or regularizing traffic across multiple websites. [6,7,11]
- Randomize website traffic to hide features in network traffic that are valid for website fingerprinting. [5,17]

Since website content is constantly changing, more latency and bandwidth overheads need to be added to make all website traffic the same. Usually low overhead website fingerprinting defense algorithms are randomized algorithms. Therefore we choose to use randomization to implement website fingerprinting defense. The traffic generated in website loading is mainly generated by HTTP requests and responses, so we analyze the association between HTTP and website traffic, and then explore the website traffic characteristics.

Table 1 shows HTTP requests and responses generated when loading the homepage of *Google*. Some of these HTTP request and response transmission times are very close or even overlap. We define HTTP requests sent in a short time period as an HTTP burst. There are 6 HTTP bursts in this website, which can be denoted as [(1),(2–5),(6),(7–9),(10),(11)]. In each HTTP burst, the sum

Table 1. HTTP requests and responses recorded when visiting https://www.google.com/ Request time indicates the time from the headers sent to the last byte sent. Response time indicates the time from the headers received to the last byte received. Time is in seconds and size is in bytes.

ID	Request Time	Response Time	Request Size	Response Size
1	0.0–0.001	0.372–0.664	422	48362
2	1.268–1.278	1.663–1.667	693	6723
3	1.276–1.278	1.667–1.674	709	352
4	1.277–1.278	1.777–1.859	687	1414
5	1.276–1.278	1.858–1.859	824	352
6	2.163–2.164	3.012–3.014	645	2304
7	3.473–3.475	3.873–3.874	501	1524
8	3.457–3.460	3.756–3.875	506	66251
9	3.556–3.557	3.872–3.986	388	1376
10	4.064–4.065	4.556–4.557	842	352
11	5.977–6.055	6.493–6.494	769	37654

of the HTTP request size determines the amount of data sent, and the sum of the HTTP response size determines the total amount of data received. We count the total size of requests and responses in an HTTP burst. The size of each HTTP burst is shown on the left-hand side of Fig. 2. The right-hand side of Fig. 2 shows the trend of the traffic rate during the loading of *www.google.com*.

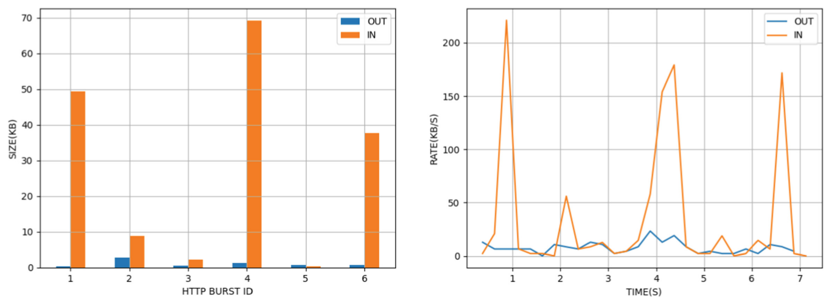

Fig. 2. The left graph shows the size of the six HTTP bursts generated when accessing *www.google.com*. The right graph shows the trend of network transfer rate when accessing *www.google.com*. It can be clearly seen that there is a correlation between the two graphs.

5 AAP

AP algorithm chooses an expected inter-packet interval(EIPI) after receiving a packet and decides whether to send a dummy packet based on EIPI. Tor developers are interested in deploying AP-style defenses because they have no latency overhead. We propose a new website fingerprinting defense, Advanced Adaptive Padding(AAP). AAP is an AP-style defense that tries to obfuscate burst patterns by sending fake bursts, it does not bring any latency overhead.

The state of AAP is shown in Fig. 3. AAP has three modes: real burst mode, fake burst mode, and gap mode.

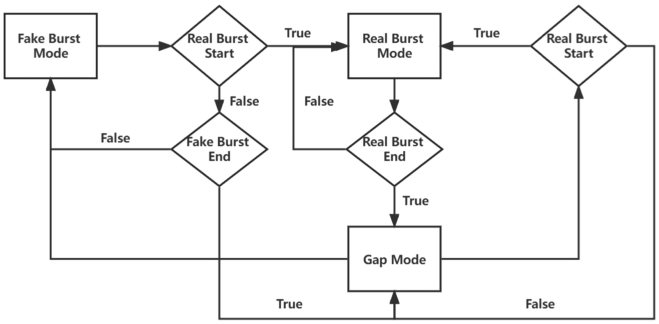

Fig. 3. Finite state machine to illustrate the AAP algorithm.

Real Burst Mode. In the real burst mode, the browser is performing network activities, such as sending HTTP requests or receiving HTTP responses. User traffic is being transmitted and the network utilization is high. Therefore, in this mode, AAP only forwards user traffic and does not send dummy packets to avoid network congestion which affects the real user traffic. AAP uses a sliding window to determine whether to be in the real burst mode. We denote the window size as W. If at least K application packets have been forwarded in the past W seconds, AAP is in the real burst mode. When the real traffic forwarding is completed, i.e., less than K packets have been sent in the past W seconds, the real burst mode ends and AAP switches to Gap mode.

Gap Mode. In the gap mode, the browser is idle, so no user traffic is currently being transmitted. AAP needs to decide when the next fake burst will be sent. For the randomness of the burst interval, AAP samples t from the distribution of real burst intervals and uses $g = t * Q$ as the time interval between two bursts. Q is a parameter greater than 0 and less than 1. Q is used to control the expectation of the time interval and has an impact on the bandwidth overhead. When AAP enters the gap mode, it immediately calculates g and starts timing. If a real burst starts before g expires, then AP transfers to the real burst mode, otherwise, it transfers to the fake burst mode.

Fake Burst Mode. In the fake mode, AAP generates a fake burst and sends it. AAP uses the feature distribution of real bursts to generate fake bursts, in order to prevent attackers from being able to distinguish the fake bursts. First, AAP randomly selects the fake burst length k from the length distribution of real bursts. Then, AAP selects $k - 1$ time intervals as the intervals between packets in the fake burst. Due to significant differences in burst characteristics between incoming and outgoing directions, it is necessary to use different distributions for each direction. During the process of sending a fake burst, AAP always detects if a real burst is sent. If so, AAP stops sending fake bursts and switches to the real burst mode. In this case, the sent dummy packets and the application packets sent later together form a large burst, changing the length of the real burst. When the fake burst is sent, AAP switches to the gap mode.

The first few seconds of each trace leak the most useful features, because the first request sent by each website is a request for an HTML file which is relatively stable and sent separately. So AAP starts in the fake mode to obfuscate the feature-rich front part of a trace. In particular, the first fake burst does not terminate early, even if it encounters a real burst.

To control the bandwidth overhead, AAP uses a parameter N to limit the total number of dummy packets. To increase the intra-class variance, an intuitive idea is to use random N for different instances of the same website. But after trying different random strategies, we find that using a fixed N is more effective and stable. As the proportion of incoming and outgoing packets is an important feature, we use a random value P to determine the ratio of outgoing packets. P is sampled from a uniform distribution between 0 and 1 for each trace.

6 Dataset

In this section, we describe the dataset collection process and the data representation in our dataset.

6.1 Data Collection

We collect a new dataset between November and December 2022 to investigate the association of HTTP requests and responses with network traffic characteristics. During access to the website, we not only capture the generated packets on the network card but also record the HTTP requests sent and responses received.

For the data collection process, we used 10 virtual machines in a cloud environment, each virtual machine is provisioned with 2 CPUs and 4GB of RAM. We select 100 popular websites from Alexa[1] as the monitored set, each visited 100 times, and 10,000 other websites as the non-monitored set, each visited once. We use selenium to control Firefox for web access. We use mitmproxy [24] to log the information of HTTP requests and responses. At the same time, we use tcpdump to capture packets on the net card and save them in pcap files. Since

[1] www.alexa.com.

the packet payloads are encrypted and thus have no value for the adversary, we extract metadata from the traffic traces and discard pcap files for saving storage. Each website is given 180 s to load before the browser is killed, and the timeout is marked as invalid. Upon loading the page, it is left open for additional 10 s, after which the browser is closed and any profile information is removed. Besides, we remove pages with request failure rates greater than 50%.

6.2 Data Representation

We follow the approach proposed by Wang and Goldberg [25] to process data and extract Tor cells from the captured pcap file. We use a sequence of cells as a traffic trace, denoted as $T = [(t_1, d_1), (t_2, d_2)...(t_{|T|}, d_{|T|})]$ where $|T|$ is the total number of cells in the trace, t_i is the timestamp of the i-th cell, d_i shows the direction of the i-th cell. The incoming and outgoing cells are represented as -1 and +1, respectively.

6.3 Ethical Consideration

Since large-scale data collection may have some impact on the Tor network, we try to mitigate the adverse effects on the Tor network. We use scripts to directly drive website visiting, so none of those visits come from real users. We just keep the minimal information that is necessary for our experiment. We only visit one web page at a time, there is no parallel processing, so for Tor, it is just one more user visiting and no additional burden.

7 Experiment Settings

To evaluate the improvements in performance offered by AAP. We use CUMUL [26], k-FP [26], DF [2] as attackers to evaluate AAP. Because these methods are state-of-the-art for different times and all achieve a high level of accuracy. CUMUL derives features from the cumulative representation of the trace and uses LibSVM with a Radial Basis Function (RBF) for classification. k-FP uses random forests to extract a fingerprint for each trace and uses KNN for classification. DF is the current state-of-the-art website fingerprinting attack algorithm, which is based on deep learning. It uses packet direction sequence as input and uses CNN for feature extraction and uses fully-connected layers for classification.

We choose four defenses, TAMARAW [11], WTF-PAD [5], FRONT [17] and WF-GAN [20] as competitors to our defenses. TAMARAW is a regularization defense with high bandwidth overhead and latency overhead. TAMARAW has high security, so it is often used as a benchmark for comparison. WTF-PAD, FRONT and WF-GAN are lightweight obfuscation defenses, they all have no latency overhead without delaying user traffic. WTF-PAD is an improvement on AP, and it is the main subject of our comparison. FRONT obfuscates the front part of the trace, it is the state-of-the-art zero-delay defense. WF-GAN is an adversarial-based defense that has very low bandwidth overhead.

We use simulation experiments to verify the effectiveness of website fingerprinting defenses. For each trace, we generate post-defense traces based on different defense protocols. Then we use attack methods to validate their defensive effect on the defended dataset. To ensure the accuracy of the results, we perform 10-fold cross-validation on the dataset. We evalute them in both closed world scenario and open world scenario.

Closed World Scenario. In the closed world scenario, the victim is restricted to access only the websites in the monitored set, the attacker's goal is to identify which website the victim has visited. So it can be seen as a multi-classification problem. The closed world scenario is an ideal scenario for testing website fingerprint attacks and defenses.

Open World Scenario. In the open world scenario, the victim can access any of the websites on the Internet, the goal of the attacker is to determine whether the website visited by the victim is in the monitored set or not. So it can be seen as a binary classification problem. The open world scenario is a relatively realistic scenario.

Metrics. Website fingerprinting defenses should be evaluated in terms of both overhead and defense effect. Overhead includes latency overhead and bandwidth overhead. Latency overhead is the additional time needed to load the website as a percentage of the original loading time. Bandwidth overhead is the ratio of the additional traffic transferred to the original website traffic. Defense effect can be shown by attack effect. In the closed world scenario, the effect of website fingerprinting is usually judged using accuracy. Accuracy is the proportion of correctly classified traces to the total number of traces. In the open world scenario, we use TPR, FPR, and F1 to judge the effect of website fingerprinting attacks. TPR is the percentage of samples in the monitored traces that are correctly classified. FPR is the percentage of samples in the non-monitored traces that are wrongly classified. F1 is the reconciled average of precision and recall.

8 Experiment Results

In this section, we provide our experiment results and compare the performance of AAP with other defenses.

8.1 Overhead

Table 2 summarizes the bandwidth overhead and latency overhead for each defense on our data set. With the exception of TAMARAW, other defenses have no latency overhead due to the fact that they do not intentionally delay sending packets. TAMARAW uses a fixed time interval to send packets, so it delays the delivery of user traffic, causing it takes too long to load websites. Its bandwidth and latency overheads are 85.92% and 73.66%, respectively, which are too high to be deployed in Tor. WTF-PAD is relatively lightweight, and it results in 66.59% bandwidth overhead. FRONT uses parameters to control the number of injected packets, with the default parameters, it incurs 54.42% bandwidth overhead. WF-GAN uses Generative Adversarial Networks to generate small perturbation. It has the lowest bandwidth overhead 6.17%. We set $N = 4000, Q = 0.4$, which makes AAP have a relatively small overhead compared to FRONT. In this setting, AAP incurs 40.37% bandwidth overhead. The parameters we used are shown in Table 2.

Table 2. Parameters and overheads of different defenses. BO for bandwidth overhead and LO for latency overhead. Overheads are all percentages.

Defenses	Parameters	BO	LO
No defense	None	0	0
TAMARAW	$\rho_{in} = 40, \rho_{out} = 12, L = 100$	85.92	73.66
WTF-PAD	$nomal\_rcv$	66.59	0
FRONT	$N_s = N_c = 2500, W_{min} = 1, W_{max} = 14$	54.42	0
WF-GAN	$\alpha = 1, \beta = 3$	8.91	0
AAP	$N = 4000, Q = 0.4$	40.37	0

8.2 Closed World

The closed world setting is useful for illustrating the effectiveness of website fingerprinting attacks and defenses. Table 3 shows how well website fingerprinting attacks perform against our evaluated defenses in the closed world setting.

Table 3. Accuracy of website fingerprinting attacks against different defenses in the closed world setting.

	CUMUL	k-FP	DF
No defense	94.55%	93.18%	97.10%
TAMARAW	16.19%	9.35%	5.19%
WTF-PAD	75.49%	74.38%	86.57%
FRONT	31.23%	59.27%	41.07%
WF-GAN	83.75%	89.17%	92.62%
AAP	**31.24%**	**47.49%**	**36.35%**

All attacks achieve high accuracy of over 93% on the undefended dataset. DF is the strongest attack since its accuracy is 97.10% and k-FP is the weakest attack with 93.18% accuracy.

The performance of AAP is worse compared to TAMARAW. However, TAMARAW has the highest bandwidth overhead and latency overhead, which makes it impossible to be deployed in practice. The other three defense schemes have cheaper bandwidth overhead and no latency overhead, but they are less effective than AAP. FRONT is the most effective method of them and has a similar overhead to AAP, but it has a lower defensive success rate against k-FP than AAP by 11.78%.

We find that AAP is less effective against k-FP and more effective against CUMUL and DF. Since AAP does not delay user traffic delivery, it leaks some time features, which k-FP can use effectively. CUMUL and DF use cumulative packet length and packet direction as features, respectively, and thus cannot exploit these time features, resulting in their poor effectiveness in combating AAP.

8.3 Open World

Table 4 shows how well website fingerprinting attacks perform against our evaluated defenses in the open world scenario.

Table 4. Defense performances in the open world scenario. A low F1 score represents a better defense. TPR and FPR are in percentage.

Defenses	CUMUL			k-FP			DF		
	TPR	FPR	F1	TPR	FPR	F1	TPR	FPR	F1
No defense	57.40	31.82	0.61	86.81	5.68	0.90	94.26	13.59	0.91
TAMARAW	12.80	21.90	0.19	52.85	47.22	0.53	23.10	82.20	0.23
WTF-PAD	50.79	33.20	0.55	52.82	7.30	0.66	70.63	23.72	0.72
FRONT	58.52	45.29	0.57	36.92	8.50	0.51	70.39	45.49	0.64
WF-GAN	54.70	42.31	0.59	87.03	9.35	0.85	92.45	82.20	0.91
AAP	**52.34**	**46.31**	**0.53**	**52.00**	**28.02**	**0.52**	**61.83**	**16.00**	**0.57**

When no defense is implemented, DF is the strongest website fingerprinting attack with a 0.91 F1 score and 94.26% TPR. CUMUL performs worst, it only has 57.40% TPR and 0.61 F1 score.

In the open world scenario, AAP can reduce the F1 score from 0.91 to 0.57 when defending against DF. AAP is still worse compared to TAMARAW, but better than the other defense schemes.

8.4 Discussion

AAP is a distribution-based defense with randomness, which produces different results for the same input. So it does not provide a theoretically provable security guarantee. In terms of defensive effectiveness, AAP is inferior to defense algorithms that can provide theoretical security guarantees, such as TAMARAW. However, these defense algorithms have a heavy overhead, which is not tolerated by Tor. As a result, these methods exist only on paper and cannot be deployed on Tor.

AAP and WTF-PAD are based on AP, but AAP outperforms WTF-PAD for many reasons. First, WTF-PAD uses packets interval to determine the interval of bursts. This approach is problematic due to the presence of background noise in the network. AAP uses a sliding window to make the distribution of bursts more reasonable. Besides, AAP uses a more random strategy, such as controlling the proportion of packets going in and out of the direction which increases intra-class variation. Finally, AAP only sends fake bursts between real bursts, when a real burst starts, it stops sending fake bursts. This avoids network congestion and improves network quality.

FRONT only obfuscates the front part of the trace, it sends a lot of noise when the website starts to load, without taking into account the impact on network congestion. The latter part of trace still gives away information. AAP only sends fake bursts when the network is idle and covers a larger area, so it works better.

Adversarial-based defenses such as WF-GAN can spoof fixed attack models with low overheads. However, when training with defended data, adversarial defenses perform poorly.

9 Conclusion

In this paper, we propose a new concept, HTTP burst, and analyze its association with network traffic. We find that HTTP burst is a key feature affecting network traffic, and based on this, we propose a new website fingerprinting defense method AAP. AAP obfuscates burst patterns by sending fake bursts in the long gap between real bursts. We evaluate AAP in both closed world scenario and open world scenario. Experiment results show that AAP outperforms other zero-delay defenses with lower overhead. AAP does not delay user traffic and it only incurs moderate bandwidth overhead, which makes it highly available to be adopted by Tor.

Acknowledgment. This work was supported inpart by the National Natural Science Foundation of China (61972219), the Research and Development Program of Shenzhen (JCYJ20190813174403598), the Overseas Research Cooperation Fund of Tsinghua Shenzhen International Graduate School (HW2021013), the Youth Innovation Promotion Association CAS (2019163), the Guangdong Basic and Applied Basic Research Foundation (2022A1515010417), the Key Project of Shenzhen Municipality(JSGG20211029095545002), the Science and Technology Research Project of Henan Province(222102210096), the Key Laboratory of Network Assessment Technology, Institute of Information Engineering, Chinese Academy of Sciences, and Shenzhen Science and Technology Innovation Commission (Research Center for Computer Network (Shenzhen) Ministry of Education).

References

1. Syverson, P., Dingledine, R., Mathewson, N.: Tor: the secondgeneration onion router. In: Proceedings of the 13th USENIX Security Symposium, (San Diego, CA, USA), pp. 303–320, USENIX Association (2004)
2. 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, CCS, (Toronto, ON, Canada), pp. 1928–1943 (2018)
3. Cherubin, G., Jansen, R., Troncoso, C.: Online website fingerprinting: evaluating website fingerprinting attacks on tor in the real world. In: 31st USENIX Security Symposium (USENIX Security 22), pp. 753–770 (2022)
4. Shmatikov, V., Wang, M.-H.: Timing analysis in low-latency mix networks: attacks and defenses. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 18–33. Springer, Heidelberg (2006). https://doi.org/10.1007/11863908_2
5. 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
6. Dyer, K.P., Coull, S.E., Ristenpart, T., Shrimpton, T.: Peek-a-boo, i still see you: why efficient traffic analysis countermeasures fail. In: 33rd IEEE Symposium on Security and Privacy, pp. 332–346 (2012)
7. 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, WPES, (Scottsdale, AZ, USA), pp. 121–130. ACM (2014)
8. 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, CCS, (Scottsdale, AZ, USA), pp. 227–238. ACM (2014)
9. Lu, D., Bhat, S., Kwon, A., Devadas, S.: DynaFlow: an efficient website fingerprinting defense based on dynamically-adjusting flows. In: Proceedings of the 2018 Workshop on Privacy in the Electronic Society, WPES, (Toronto, Canada), pp. 109–113. ACM (2018)
10. 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)
11. Wang, T., Cai, X., Nithyanand, R., Johnson, R., Goldberg, I.: Effective attacks and provable defenses for website fingerprinting. In: Proceedings of the 23rd USENIX Security Symposium, (San Diego, CA, USA), pp. 143–157, USENIX Association (2014)
12. Wang, T., Goldberg, I.: Walkie-Talkie: an efficient defense against passive website fingerprinting attacks. In: Proceedings of the 26th USENIX Security Symposium, (Vancouver, BC), pp. 1375–1390, USENIX Association (2017)
13. Panchenko, A., Niessen, L., Zinnen, A., Engel, T.: Website fingerprinting in onion routing based anonymization networks. In: Proceedings of the 10th annual ACM workshop on Privacy in the electronic society, WPES, (Chicago, IL, USA), pp. 103–114 (2011)

14. Luo, X., Zhou, P., Chan, E.W., Lee, W., Chang, R.K., Perdisci, R., et al.: HTTPOS: sealing information leaks with browser-side obfuscation of encrypted flows. In: Proceedings of the Network and Distributed System Security Symposium, NDSS, (San Diego, CA, USA) (2011)
15. Cherubin, G., Hayes, J., Juárez, M.: Website fingerprinting defenses at the application layer. Proc. Priv. Enhan. Technol. **2017**(2), 186–203 (2017)
16. 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, CCS, pp. 605–616. ACM (2012)
17. Gong, J., Wang, T.: Zero-delay lightweight defenses against website fingerprinting. In: Proceedings of the 29th USENIX Security Symposium, (Boston, MA, USA), pp. 717–734, USENIX Association (2020)
18. Henri, S., García, G., Serrano, P., Banchs, A., Thiran, P., et al.: Protecting against website fingerprinting with multihoming. Proc. Priv. Enhan. Technol. **2020**(2), 89–110 (2020)
19. 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, CSS, (Virtual Event, USA), pp. 1971–1985. ACM (2020)
20. Hou, C., Gou, G., Shi, J., Fu, P., Xiong, G.: WF-GAN: fighting back against website fingerprinting attack using adversarial learning. In: IEEE Symposium on Computers and Communications, ISCC, (Rennes, France), pp. 1–7. IEEE (2020)
21. Xiao, C., Li, B., Zhu, J.-Y., He, W., Liu, M., Song, D.: Generating adversarial examples with adversarial networks. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI, pp. 3905–3911. AAAI (2018)
22. Rahman, M.S., Imani, M., Mathews, N., Wright, M.: Mockingbird: defending against deep-learning-based website fingerprinting attacks with adversarial traces. IEEE Trans. Inf. Forensics Secur. **16**, 1594–1609 (2020)
23. Nasr, M., Bahramali, A., Houmansadr, A.: Defeating DNN-based traffic analysis systems in real-time with blind adversarial perturbations. In: Proceedings of the 30th USENIX Security Symposium, (Vancouver, B.C., Canada), pp. 2705–2722, USENIX Association (2021)
24. Cortesi, A., Hils, M., Kriechbaumer, T.: mitmproxy: a free and open source interactive HTTPS proxy (2010) [Version 9.0]
25. Wang, T., Goldberg, I.: Improved website fingerprinting on tor. In: Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society, WPES, (Berlin, Germany), pp. 201–212. ACM (2013)
26. Panchenko, A., et al.: Website fingerprinting at internet scale. In: Proceedings of Internet Society Symposium on Network and Distributed Systems Security, (San Diego, CA, USA). IEEE (2016)