Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.049504

ARTICLE

CMAES-WFD: Adversarial Website Fingerprinting Defense Based on Covariance Matrix Adaptation Evolution Strategy

Di Wang, Yuefei Zhu, Jinlong Fei* and Maohua Guo

School of Cyberspace Security, Information Engineering University, Zhengzhou, 450000, China

*Corresponding Author: Jinlong Fei. Email: feijinlong_2021@163.com

Received: 09 January 2024 Accepted: 13 March 2024 Published: 15 May 2024

ABSTRACT

Website fingerprinting, also known as WF, is a traffic analysis attack that enables local eavesdroppers to infer a user's browsing destination, even when using the Tor anonymity network. While advanced attacks based on deep neural network (DNN) can perform feature engineering and attain accuracy rates of over 98%, research has demonstrated that DNN is vulnerable to adversarial samples. As a result, many researchers have explored using adversarial samples as a defense mechanism against DNN-based WF attacks and have achieved considerable success. However, these methods suffer from high bandwidth overhead or require access to the target model, which is unrealistic. This paper proposes CMAES-WFD, a black-box WF defense based on adversarial samples. The process of generating adversarial examples is transformed into a constrained optimization problem solved by utilizing the Covariance Matrix Adaptation Evolution Strategy (CMAES) optimization algorithm. Perturbations are injected into the local parts of the original traffic to control bandwidth overhead. According to the experiment results, CMAES-WFD was able to significantly decrease the accuracy of Deep Fingerprinting (DF) and VarCnn to below 8.3% and the bandwidth overhead to a maximum of only 14.6% and 20.5%, respectively. Specially, for Automated Website Fingerprinting (AWF) with simple structure, CMAES-WFD reduced the classification accuracy to only 6.7% and the bandwidth overhead to less than 7.4%. Moreover, it was demonstrated that CMAES-WFD was robust against adversarial training to a certain extent.

KEYWORDS

Traffic analysis; deep neural network; adversarial sample; Tor; website fingerprinting

1 Introduction

Website fingerprinting attack is a traffic analysis attack that uses unique identifying features of websites, such as packet length, packet direction, and packet interval, to infer a user's browsing targets. Privacy Enhancing Technologies (PETs), such as Virtual Private Network (VPN) and The Onion Router (Tor), have been adopted to protect user privacy [1]. Tor also encrypts data and hides information about the size of the packets by encasing them in fixed-size cells. However, time and direction information are still accessible, allowing attackers to construct unique WF features and infer users' browsing targets based on this information.

The attackers utilized traditional machine learning in the early stages of WF attacks. The three optimal attack methods were CUMUL [2], K-Fingerprinting (K-FP) [3], and K Nearest Neighbors (K-NN) [4], all achieving over 90% accuracy. However, the performance of traditional machine learning methods depends on manually selected features and classifiers. Over the past few decades, DNN has made great achievements in image classification and speech recognition. Inspired by this, many researchers have tried using DNN for WF attacks and achieved significant results, such as VarCnn [5], and DF [6] have achieved over 98% accuracy. DF can achieve 90% accuracy even for traffic protected by defensive methods like WTF-PAD [7]. Moreover, unlike machine learning, DNNs can perform feature engineering automatically without requiring manual calculation and feature selection. Numerous studies [8–11] have demonstrated that, even though the DNN-based approaches have great advantages, DNNs are vulnerable to adversarial samples: Carefully crafted inputs with small perturbations can cause DNN classifiers to misclassify. Szegedy et al. [12] first observed adversarial examples in image classification, where small changes to an image can fool the classifier, typically with imperceptible changes to human eyes. The viability of using adversarial examples in WF defense has been the subject of some research [13–17]. DNN classifiers can be fooled by selecting a target website and morphing the traffic patterns of the source website into those of the target website [14,17]. However, this method depends on the chosen target website; if the traffic pattern of the source website is significantly different from that of the target website, there will be an unacceptable bandwidth overhead. Other studies [15,18] dedicate to generating adversarial perturbations that can be used for real-time traffic. However, these methods require access to the loss function or internal parameters of the target models [13]. The target model used by attackers is unknown to us, so this white-box-based defense approach does not fit realistic scenarios. The study also uses an adversarial patch to perturb website traffic [15]. However, the generated adversarial patch is trace-oriented. It may not work for different traces from the same website and may cause a lot of new bursts, increasing time and bandwidth overhead.

This paper proposes CMAES-WFD, a black-box defense method based on adversarial samples. Perturbation generation and perturbation injection are the two main parts of CMAES-WFD. CMAES-WFD transforms the generation of perturbations into a constrained optimization problem. By random sampling from the Gaussian distribution, the generation of perturbation uses feedback from the target model to solve the constrained optimization problem. The perturbation injection adds bound constraints and only selects several critical positions to insert dummy packets. CMAES-WFD is website-oriented. Therefore, the perturbation can be applied to other traces of the same website. Experiments show that CMAES-WFD effectively balances accuracy and bandwidth overhead and is also effective in resisting adversarial training. In summary, our main contributions are as follows:

(1) We propose a WF defense based on adversarial sample. CMAES-WFD uses a black-box optimization method to generate perturbation without accessing the model's structure and internal parameters. We defined minimizing website classification accuracy as the optimization objective, solved using the CMAES. CMAES does not require the objective function's explicit analytical forms, nor is the gradient's calculation necessary for solving. Perturbation is randomly sampled from a Gaussian distribution, and due to the random nature of the sampling, the generated perturbation was robust against adversarial training to a certain extent.

(2) CMAES-WFD is website-oriented, and perturbation can be applied to various traces of the same website. Typically, dummy packets cannot be injected into trace when the data transmission ends, so a boundary constraint is added to the perturbation injection function. When injecting perturbation, we selected a few positions with high variance based on the

CMAES updating principle of increasing variance along successful search directions. This method reduced bandwidth overhead while simultaneously increasing robustness.

(3) Through various experiments, we evaluate CMAES-WFD and discover that it made a trade-off between accuracy and bandwidth overhead, better than most previous defenses. Furthermore, our method demonstrated strong generalization ability and the adversarial perturbation was transferable among different models, making it effective against unknown attack models.

The rest of the paper is organized as follows. In Section 2, we review previous studies on WF attacks and defenses. In Section 3, we describe the preliminaries of the investigated problem. In Section 4, we introduce our WF defense named CMAES-WFD. In Section 5, we describe the experimental setup, while Section 6 shows the experimental results. Section 7 discusses the limitation of the paper and future work. In Section 8, we conclude the paper.

2 Related Work

This chapter mainly reviews the classic network WF attack and defense methods so far.

Traditional ML-Based WF Attacks: Early WF attacks mostly used manual feature engineering-based traditional machine learning models. Panchenko et al. [2] proposed CUMUL based on Support Vector Machine (SVM). They used statistical features and interpolated features sampled from the cumulative representation of directional packet length sequences, and achieved over 91% accuracy in closed-world scenarios; Hayes et al. [3] proposed K-FP attack. They fed traditional packet features into a random forest and used the output of leaf nodes as the final feature representation, and then input the new features into K-NN, achieving 91% accuracy in the closed-world scenario; Wang et al. [4] used K-NN classifier and collected various features including packet order, incoming and outgoing packet counts, and bursts. In the closed-world scenario, they achieved 91% accuracy by combining the features and employing a distance metric to measure the similarity among websites; Hermann et al. [19] conducted the first WF study with Multinomial Navie Bayers (MNB), achieving only 3% accuracy in Tor's scenario with 775 websites. Based on this, Panchenko et al. [20] continuously increased the variety of features used and denoised the data to increase the accuracy to about 55% using SVM.

DNN-Based WF Attacks: However, the effectiveness of attacks [19–22] based on traditional machine learning largely depends on the classifier and features chosen. Inspired by the excellent performance of DNNs in image classification and speech recognition, numerous studies have used DNNs to perform more effective WF attacks, which take the packet direction sequence of the raw traffic as input and automatically conduct feature engineering. Table 1 shows the comparison between machine learning-based and deep learning-based WF attacks. Bhat et al. [5] proposed VarCnn, a more complex WF attack based on ResNet-18 architecture that achieves nearly 99% accuracy in a closed-world scenario and further enhances attack performance; Sirinam et al. [6] proposed a new CNN-based attack named DF. Compared to AWF, DF has more convolutional layers for extracting traffic features and achieves over 98% accuracy in closed-world scenarios, outperforming all previous attack methods. Additionally, with an accuracy of over 90%, DF remains useful for WTF-PAD defense; Abe et al. [23] were the first to apply DNN to WF attacks. They proposed an attack method based on the Stacked Denoising Autoencoder (SDAE). It only achieved 88% accuracy, which was lower than the accuracy of advanced machine learning-based attacks. However, SDAE first used sequence of packet direction as traffic feature, with -1 and $+1$ representing incoming and outgoing packets, respectively [13]; Rimmer et al. [24] proposed Automated Website Fingerprinting (AWF), a deep learning-based attack for automatic feature engineering. They compared the feature extraction capabilities of three different architectures-SDAE, Convolutional Neural Network (CNN), and Long-Short Term Memory (LSTM). The result shows that CNN performs best, achieving 96% accuracy after training on 100 websites and 2500 traces per website.

Table 1: WF attacks based on traditional learning or deep learning

Attack	Classifier	Weakness
CUMUL [2]	SVM	Manual feature extraction and the classification performance depends on the classifier and features chosen
K-FP [3]	Random forest	
K-NN [4]	k nearest neighbors	
MNB [19]	Multinomial Navie Bayers	
VarCnn [5]	ResNet	Requires a lot of training data
DF [6]	CNN	
SDAE [23]	Autoencoder	
AWF [24]	CNN	

WF Defenses: To defend against WF attacks, researchers have proposed various defenses to hide traffic characteristics. Juarez et al. [7] improved the Adaptive Padding (AP) defense and proposed an adaptive WF defense named WTF-PAD. WTF-PAD sample intervals from a predefined histogram of packet arrival time distribution, and send dummy packets within the sampled interval when there are no real packets in the buffer. WTF-PAD can effectively resist WF attacks based on traditional machine learning but weakens against attacks based on deep learning [25]; Dyer et al. [26] proposed the Buffered Fixed Length Obfuscator (BuFLO), which protects against traffic analysis attacks by concealing side-channel data using fixed-length, fixed-rate, and fixed-interval modes. However, BuFLO is expensive for websites with small packet sizes and short loading times and cannot adjust to changes in network speed; Cai et al. [27] aimed to make BuFLO more practical by proposing Tamaraw to reduce BuFLO's overhead. Furthermore, Cai et al. [28] proposed the Congestion-Sensitive BuFlO (CS-BuFLO), which can adapt to various network speeds and enhances BuFLO's rate adaptation and packet padding mechanisms; Wang et al. [29] proposed Walkie-Talkie (W-T), which finds a super-sequence for sensitive and non-sensitive pages by treating them as a group. However, W-T requires the client and browser to communicate in half-duplex mode and requires modifying the way the browser loads the page; Abusnaina et al. [30] proposed Deep Fingerprinting Defender (DFD), which mainly consists of two modules: Burst observer and injection buffer. Burst observer is used to record the length of the last burst and calculate the number of dummy packets accordingly. Injection buffer injects dummy packets into real-time traffic. DFD reduces accuracy to 14% with 14.2% bandwidth overhead by client-side injection.

WF defenses based on adversarial samples have been adopted by numerous researchers in recent years. The comparison between traditional defense methods and adversarial sample-based defense methods is shown in Table 2. As mentioned earlier, numerous studies [31–34] have demonstrated that DNNs are vulnerable to adversarial samples despite the distinct advantages of DNN-based WF attacks. As a result, efforts to use adversarial samples in WF defenses have been made, and significant progress has been made in achieving a balance between accuracy and bandwidth overhead. Li et al. [13] proposed MiniPatch, a lightweight WF defense. MiniPatch calculates the size of the adversarial patches and the insertion position by simulated annealing algorithm, and it reduces model's accuracy to 3% with less than 5% bandwidth overhead; Sadeghzadeh et al. [14] proposed Adversarial

Website Adaption(AWA), an adversarial website adaptation defense based on GAN. AWA randomly divides websites into pairs, both source and target websites, and trains a transformer for each website so that the burst sequences of the two websites are constantly close to each other. AWA can reduce the accuracy of DF with adversarial training to 19.5% with 22.3% bandwidth overhead. Moreover, AWA can generate universal perturbations without accessing a website's trace; Shan et al. [15] proposed a patch-based WF defense by injecting pre-computed adversarial patches into the network traffic, which can also be used for real-time traffic. Dolos selects a target website trace from a pool of possible candidates, and the adversarial patch is determined by minimizing the l2 distance between the original traffic and the target traffic features in DNN space. However, this process requires white-box access to the target model. Dolos implements segment-based patch injection, dividing the adversarial patch equally and injecting each patch segment into a specific location of the original trace. This may introduce new bursts if inject reverse packets. Even though Dolos reduces DF's accuracy to less than 6%, it still requires a bandwidth overhead of 30%; Inspired by AdvGAN, Hou et al. [16] proposed a WF defense based on Generative Adversarial Network (GAN). WF-GAN takes the burst sequences as input and guides the generator to generate perturbations using the discriminator's loss function and the target model's output. With less than 15% bandwidth overhead, WF-GAN reduces DF's accuracy to around 10%; Rahman et al. [17] first applied the idea of adversarial samples to WF defense by mimicking other website traffic patterns. Mockingbird selects the target trace from a candidate pool closest to the original trace regarding L2 distance, reducing DF accuracy to around 30% with 58% bandwidth overhead; Nasr et al. [18] proposed an adversarial perturbation generation approach that does not rely on specific website traces, which can be applied to real-time traffic. Blind reduces the accuracy of DF to 9% with 11% bandwidth overhead and 5% with 25% bandwidth overhead. However, training the generator in Blind requires the loss function of the target model, which is unrealistic in a black-box attack scenario.

Table 2: WF defenses with different defensive rules

Defense	Defensive rules	Weakness
WTF-PAD [7] BuFLO [26] Tamaraw [27] CS-BuFLO [28] W-T [29] DFD [30]	The rules are pre-set, such as inserting dummy packets or increasing delay regularly	Bandwidth overhead is high and the defense effect is poor against WF attack based on deep neural network
MiniPatch [13] AWA [14] Dolos [15] WF-GAN [16] MockingBird [17] Blind [18]	The packet injection rules learned against WF attacks, based on adversarial samples	Difficult to apply to practical scenarios

3 Preliminary

Threat model: We assume users use Tor to protect their online activity while browsing websites. However, even if Tor uses three different encrypted routing nodes to protect users' privacy, attackers can still extract features from the user's network traffic, feed these features into a trained classifier, and identify the user's browsing target based on the classifier's output. As can be seen in Fig. 1, this paper employs the same threat model as previous attacks [5,6,24] and defenses [17,18,35,36]. The attacker is between the user and the Tor entry node and can be a local system administrator, an Internet Service Provider (ISP), or an autonomous system. Specifically, the attacker cannot modify, drop, or decrypt packets; they can only record network traffic. We assume that we can get black-box feedback from the target model, that is, we input traffic feature into the target model and can know which website label the traffic belongs to. We further assume that the user only visits one page at a time, allowing the attacker to distinguish between the beginning and the end of the page [35]. This makes it a more arduous task for defense since multi-label browsing attacks are complex.

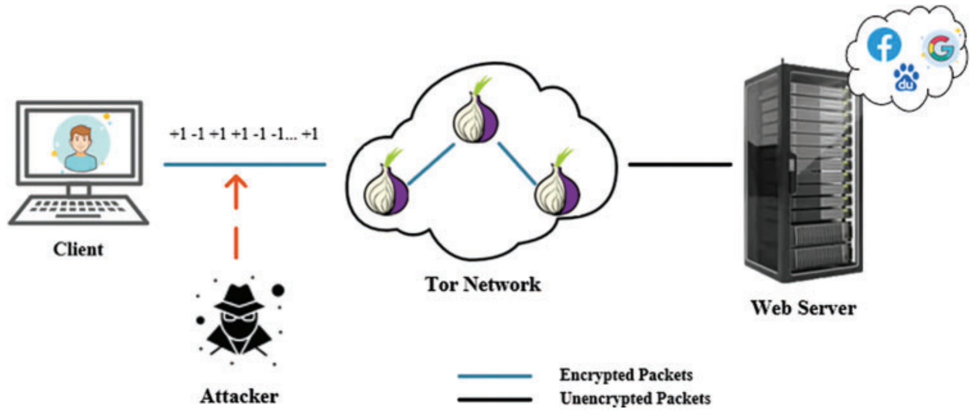


Figure 1: WF threat model

Attackers can collect network traces and extract each website's unique traffic patterns, including packet timings and directions, to train their classifiers. It is impractical to visit every website on the internet to collect traces. Attackers can collect only those sites that interest them, called the monitored set. This way, the attack scope is limited to the websites in the monitored set. Other websites are called the unmonitored set.

Closed-World and Open-World Scenarios: WF attacks and defenses [35–37] are typically evaluated in both closed-world and open-world scenarios. In the closed-world scenario, users only visit sites in the monitored set, and the attacker only collects traces from monitored sites when training their classifier. Users can visit websites from monitored and unmonitored sets in the more realistic open-world scenario, and the attackers' training data for their classifiers comes from monitored or unmonitored sites. Therefore, in the open-world scenario, an adversary needs to identify whether the websites visited by users belong to the monitored set and which class of the monitored set. In fact, the number of websites visited by users is much larger than those in the closed-world scenario. Consequently, attackers cannot fully evaluate the actual performance of their classifiers because the closed-world scenario is a relatively ideal environment for them to operate in. However, from a defense perspective, the attacker's attack ability is stronger in the closed-world scenario, so this scenario is sufficient to evaluate the defense performance. The defense effect in the closed-world scenario is the sole focus of this paper.

Data Representation: CMAES-WFD is based on burst sequences, just like some previous studies [14,16,17]. A burst is defined as a continuous packet sequence with the same direction. -1 and $+1$ indicate the direction, where $+1$ represents the client to server, and -1 represents server to client. Generally, the number of bursts for each trace is different. Like Mockingbird [17], we use the fixed length of 750 for each website trace. If traces are less than 750 bursts, they are padded with 0. If traces are more than 750, they are truncated.

However, advanced DNN-based WF attacks [5,6,24] take packet direction sequences as input. In order to be consistent with the inputs of WF attack literature, we use packet direction sequences as traffic features and feed them into the classifier. We use a fixed length of 5000 for each website trace, like DF [6]. If traces are less than 5000 packets, they are padded with 0. If traces are more than 5000, they are truncated.

Fig. 2 depicts the traffic representation in burst sequences and packet sequences. The following steps must be taken to convert between burst and packet sequences using our method: When converting packet sequences into burst sequences, packets with the same direction are accumulated, and the total accumulation is the size of the burst, with the burst direction consistent with the direction of the packets and when converting burst sequences to packet sequences, each burst is decomposed into -1 or $+1$.

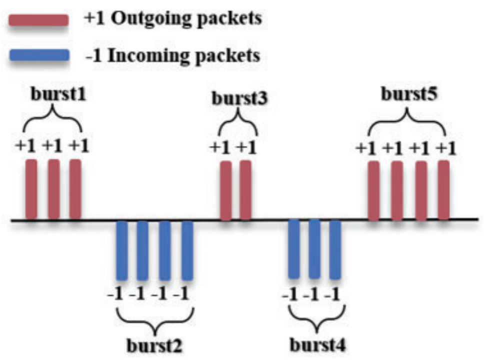


Figure 2: Burst sequences and packet sequences

Adversarial Sample: Szegedy et al. [12] first proposed adversarial samples. By adding imperceptible and small perturbations to an input, the classifier gives an incorrect output with high confidence, and the perturbed input is called an adversarial sample. The core idea of an adversarial sample is to add perturbations to the original sample, causing it to be misclassified as another class, and the size of the perturbations is limited. Specifically, given an input sample x, a trained model f, and a target class $t(t \neq f(x))$. The goal is to find a perturbation δ such that $x' = x + \delta$ $(f(x') = t)$, and x' is as close to x as possible, which can be measured by distance metrics such as l2 distance. For targeted attacks, t is a pre-specified class; for non-targeted attacks t may belong to any class other than $f(x)$. More specifically, taking non-targeted attacks as an example, the perturbation δ can be computed by solving the following optimization problem:

$$\delta = \arg{}_{\delta}^{min}[f(x+\delta) \neq f(x)]$$

$$\textit{subject to } \|\delta\|_2 < \epsilon \tag{1}$$

where ϵ is a small constant that limits the perturbation δ. Researchers have proposed various methods to generate adversarial samples in image adversarial attacks. Goodfellow et al. [38] proposed the Fast Gradient Sign Method (FGSM), which calculates the perturbation by taking the gradient of the loss function concerning the input. Papernot et al. [39] proposed the Jacobian-based Saliency Map Attack (JSMA), a local pixels attack. JSMA forms a Jacobian matrix by taking the forward derivative of the model's output for each pixel, calculates the saliency map value of each pixel based on the Jacobian matrix, and selects the pixels with the highest saliency map value to perturb; Moosavi-Dezfooli et al. [40] proposed Universal Adversarial Perturbation (UAP), which aims to generate a universal perturbation independent of a specific input by employing the shortest distance vector between the adversarial sample and the original sample as the perturbation vector. The adversarial samples are regarded as an attack in adversarial machine learning. However, for WF defense, it can be regarded as a means of defending against the attacker's classifier. Since WF defense aims to prevent attackers from identifying the user's browsing target, this article only considers non-targeted defense, where the classifier misclassifies the perturbed trace and does not need to classify it as a specific class.

4 Methodology

This section focuses on the CMAES-WFD method. After providing an overview of CMAES-WFD, which presents the generation of adversarial perturbations as a constrained optimization problem, this section provides in-depth descriptions of two crucial modules: Generating perturbation and injecting perturbation. Fig. 3 depicts the CMAES-WFD defense flowchart.

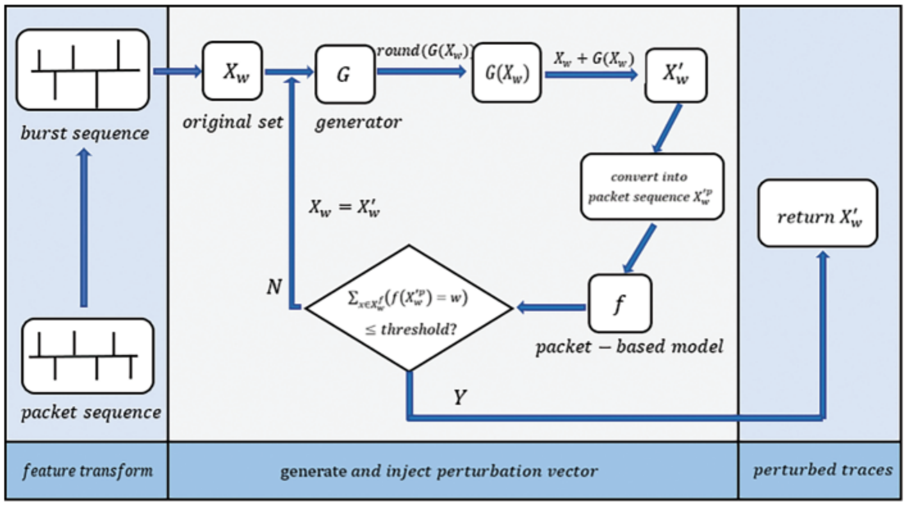


Figure 3: Flowchart of the CMAES-WFD defense

4.1 CAMES-WFD Overview

For website fingerprinting defense, the two goals of adversarial samples are respectively to reduce the classification accuracy of the model and to reduce bandwidth overhead as much as possible. Therefore, we can take reducing the classification accuracy of the model as an objective function and limiting the overhead within a certain range as a constraint condition, thus describing the generation of adversarial samples as a constrained optimization problem. When generating perturbation, burst sequences are utilized as the trace feature. However, as Section 2 describes, a conversion between packet and burst sequences is required because DNN models use packet sequences as input. Additionally, shorter traces are padded with 0, and longer traces are truncated to meet the DNN model's fixed-length input requirement and to unify the traces with different lengths.

We assume that the target model is f, and f is a DNN model that has been trained. For instance, $x(x \epsilon w)$ belongs to website w, f can correctly classify it as $w(f(x) = w)$, and $f_w(x)$ represents the confidence that x is classified as website w. Our goal is to compute the perturbation $\delta = (\delta_1, \delta_2, \delta_3, \ldots, \delta_{n_{burst}})$ and inject perturbation δ to x to obtain the adversarial sample $x'(x' = x + \delta)$ so that $f_w(x')$ decreases until x' is misclassified as w' $(f(x') = w', w' \neq w)$ or the maximum number of iterations is satisfied. Since this paper only focus on the untargeted defense, w' can be any class other than w. Therefore, the perturbation δ generation can be described as solving the following optimization problem:

$$\delta_x = \underset{\delta}{^{argmin}} (f_w(\Phi(x, \delta))$$
$$\textit{subject to}\, 0 \leq l_i \leq L\, (i = 1, 2, 3, \ldots, \gamma) \tag{2}$$

where l_i denotes the insertion position of δ_i, $\Phi(x, \delta)$ defines the injection function for injecting the perturbation to the original trace, and L denotes the transmission end boundary of instance x, $L = \min(L_x, n_{burst})$, L_x denotes the actual burst length of instance $x(x \epsilon w)$, and $n_{burst} = 750$. However, the optimization problem is trace-oriented. Even if traces are from the same website, the packet sequences may differ, and the perturbation calculated for a single trace may not have much effect on other traces from the same website. Therefore, we modified the optimization problem to transform it to be website-oriented:

$$\delta_w = \underset{\delta}{^{argmin}} \frac{1}{|X_w^f|} \sum\nolimits_{x \in X_w^f} (f(\phi(x, \delta)) = f(x)) \tag{3}$$

where $\left|X_w^f\right|$ represents traces correctly classified as w by classifier f, its size is strictly less than $|X_w|$. Injection boundary L_x takes the minimum actual length of X_w^f. In addition, we define a classification accuracy threshold θ to control the degree of optimization. Therefore, the optimization objective becomes:

$$\sum\nolimits_{x \in X_w^f} (f(\phi(x, \delta)) = f(x)) \leq \left|X_w^f\right| * \theta \tag{4}$$

The value of θ has a direct impact on defensive performance. When θ is small, the optimization requirement is stricter, which may result in lower accuracy. However, the corresponding bandwidth overhead will be higher, which can be achieved by modifying θ to compromise bandwidth overhead and accuracy.

4.2 Adversarial Perturbation Injection

Most WF defense [15,18] based on packet sequence inject the perturbations directly into specific locations of the traces, which can compromise the integrity of the data transmission. For instance, injecting a dummy packet in the opposite direction of the burst can split the complete burst into two parts and introduce new bursts. CMAES-WFD is based on bursts sequence. The burst sequence considers the data transmission direction and the burst length related to the data transmission volume so that the burst-based defense can fully use the traffic characteristics. Additionally, the direction of the dummy packets injected is the same as the bursts. The perturbation injection example is shown in Fig. 4.

CMAES-WFD selects only top γ positions with larger variances each time to reduce bandwidth overhead. The updated principle of CMAES is to increase the variance along the successful search direction. This means that the greater the variance, the more significant the position is and how much of an impact it has on the outcomes. Therefore, we set the perturbations at the remaining positions to zero, keeping only the perturbations on the top γ positions with larger variances. This measure reduces the bandwidth overhead (corresponding to lines 5–15 in Algorithm 1). In addition, since no more dummy packets can be injected once the data transmission is over, we set a limit on the injection boundary, taking the minimum true burst sequence length of traces belonging to the website w as the injection boundary. No packets will be injected at a certain position even if the variance there is large but exceeds the boundary limit (corresponding to lines 6–11 in Algorithm 1). The perturbation injection function is described in detail in Algorithm 1.

Algorithm 1: Injecting perturbation

```
Input:  $X_w$–website traces correctly classified by f as w
        $\delta$– perturbation vector
        $\gamma$–number of positions with the large variance
Output: $X_w'$–perturbed traces
 1. $L \leftarrow$ take the minimum value of the $X_w$ real lengths as the bound
 2. $C_{sort} \leftarrow$ sort the diagonal elements of covariance matrix C in descending order and return index
 3. count = 0
 4. $sign\_traces \leftarrow$ sign ($X_w$)
 5. for position index i in $C_{sort}$ do
 6.     if count < $\gamma$ then
 7.         if i < $L$ then
 8.             count = count + 1
 9.         else
10.             $\delta_i = 0$
11.         end if
12.     else
13.         $\delta_i = 0$
14.     end if
15. end for
16. $X_w' \leftarrow |X_w| + |\delta|$                |...| represents the absolute value
17. $X_w' \leftarrow X_w' * sign\_traces$
18. return $X_w'$
```

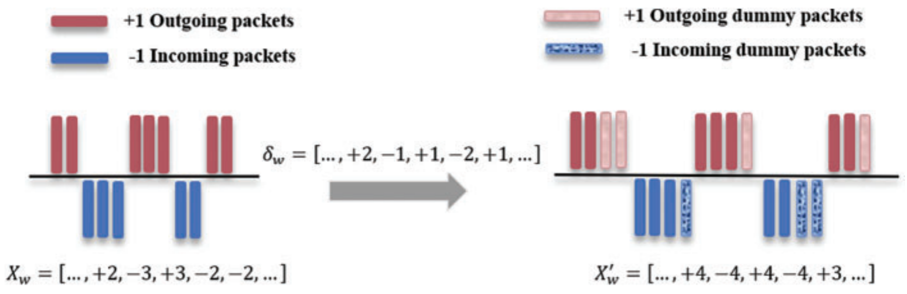

+1 Outgoing packets
-1 Incoming packets
+1 Outgoing dummy packets
-1 Incoming dummy packets
$\delta_w = [\ldots, +2, -1, +1, -2, +1, \ldots]$
$X_w = [\ldots, +2, -3, +3, -2, -2, \ldots]$
$X_w' = [\ldots, +4, -4, +4, -4, +3, \ldots]$

Figure 4: CMAES-WFD perturbation injection example

4.3 Adversarial Perturbation Generation

The objective of the CMAES-WFD is to reduce the confidence of the true label, in contrast to studies that focus on the loss function of the model [16,18] or the l2 norm distance between features

[15,17]. We cannot directly acquire specific information about the target model. Given the input, looking at the output is the simplest and most direct method to probe target model. CMAES-WFD uses the model's black-box feedback for the label to solve the optimization problem without requiring an understanding of the model's architecture or internal parameters.

This paper applies a new evolution algorithm to solve the optimization problem, namely CMAES [41]. CMAES does not require the objective function to be differentiable. Thus, it can be used for probabilistic labeling optimization. CMAES-WFD is based on a variant of CMAES–(1, λ)-CMAES. (1, λ)-CMAES generates λ offspring as candidate solutions at each iteration and selects the offspring with the highest fitness function value from the λ offspring as the parent for the next iteration. CMAES uses Gaussian distribution to search for candidate solutions randomly. The Gaussian distribution has the highest entropy of all distributions in $\mathfrak{R}^n$ given variances and covariances, and coordinate directions are not distinguished in any way. Both make the Gaussian distribution a particularly attractive candidate for randomized search [41].

The specific perturbation generation function is detailed in Algorithm 2. Perturbation generation follows the optimization problem defined in Eq. (2) and the criteria defined in Eq. (3). In each iteration, we randomly sample perturbation δ ($\delta = (\delta_1, \delta_2, \ldots, \delta_{n_{burst}})$) from the Gaussian distribution $(m, \sigma^2 C)$. CMAES-WFD is based on burst sequences, and each element δ_i of the perturbation δ represents the number of dummy packets to be injected. Therefore, the perturbation vector δ should be an integer sequence, but random sampling from the Gaussian distribution does not satisfy this criterion, and we use rounding to obtain an integer sequence δ (corresponding to lines 5–6 in Algorithm 2). The perturbation vector δ sampled from the Gaussian distribution is injected into the website traffic X_w^f to obtain an offspring (candidate solution). After λ samples, λ offspring (candidate solutions) are obtained and then calculate fitness function values of λ offspring, which in this paper is the model's accuracy for traces belonging to website w. In this iteration, we choose the candidate solution with the lowest fitness value as the best one and use it as the parent for the next iteration (lines 4–12 in Algorithm 2) until either Eq. (3) is satisfied or the maximum number of iterations is reached. The adaptive updating procedure for some parameters during each iteration is then discussed.

Adaptive updating of covariance matrix C: The update of covariance matrix C can simulate local search directions. This paper employs the rank-one update strategy, in which only the best offspring is used for updates each time. Additionally, the evolution path p_C is constructed to preserve sign information and make use of historical search information [41]. The update formula of p_C is as follows:

$$p_C^{(g+1)} = (1 - c_C)\, p_C^{(g)} + \sqrt{c_C(2 - c_C)\mu_{eff}}\,\frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}} \tag{5}$$

where $p_C^{(g)}$ denotes the evolution path of C at generation g, and c_C denotes the learning rate for cumulation for the rank-one update of covariance matrix C, and μ_{eff} denotes variance effective selection mass, and $m^{(g)}$ denotes the mean value of the distribution at generation g, and $\sigma^{(g)}$ denotes the step size at generation g [41]. The opposite direction components among these directions cancel each other out, while the same components are added up. Moreover, exponential smoothing is introduced, which assigns recent generations a higher weight. The rank-one update of covariance matrix C is:

$$C_{ii} = (1 - c_1)\, c_{ii} + c_1\, (p_C)_i^2 \tag{6}$$

where c_{ii} is the diagonal element of C, and c_1 denotes the learning rate for the rank-one update of covariance matrix C, and $(p_C)_i$ denotes the i-th element of p_C.

Adaptive updating of step size σ: CMAES adapts Cumulative Step size Adaptation (CSA) to update step size σ. The parameter σ is seperated from the covariance matrix and controls the overall scale of distribution. Parameter updates are accelerated when the step size is increased. Similar to the covariance matrix update, evolution path P_σ is constructed for step size update to determine whether the current step size is appropriate. By comparing the evolution path P_σ with the expected length $\mathrm{E}\,\|\mathcal{N}(\mathbf{0}, I)\|$ in the random selection state, if the evolution path is shorter than the expected length, reduce σ; otherwise, increase σ [41]. The update formula of σ is as follows:

$$p_\sigma^{(g+1)} = (1 - c_\sigma)\, p_\sigma^{(g)} + \sqrt{c_\sigma (2 - c_\sigma) \mu_{eff}}\, C^{(g)^{-\frac{1}{2}}}\, \frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}} \tag{7}$$

$$\sigma^{(g+1)} = \sigma^{(g)} exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma^{(g+1)}\|}{E\,\|\mathcal{N}(0, \mathfrak{I})\|} - 1\right)\right) \tag{8}$$

$$\mathrm{E}\,\|\mathcal{N}(0, \mathrm{I})\| = \frac{\sqrt{2}\,\Gamma\left(\frac{n+1}{2}\right)}{\Gamma\left(\frac{n}{2}\right)} \approx \sqrt{n} + \mathrm{O}(1/n) \tag{9}$$

where $p_\sigma^{(g)}$ denotes the evolution path of σ at generation g, and c_σ denotes the learning rate for cumulation for σ, d_σ denotes the damping parameter, and $\mathrm{E}\,\|\mathcal{N}(\mathbf{0}, I)\|$ denotes the expectation of the Euclidean norm of $\mathcal{N}(\mathbf{0}, I)$ distributed random vector [41].

Adaptive updating of mean *m*: Mean value determines the search area of the distribution. The new mean of the search distribution is obtained by averaging the μ best offspring among λ offspring. The update formula of m is as follows:

$$m^{(g+1)} = \sum\nolimits_{i=1}^{\mu} \omega_i \delta_{i:\lambda}^{(g+1)} \tag{10}$$

$$\omega_i = 1/\mu\ i = 1, 2, 3 \ldots \mu \tag{11}$$

$\delta_{i:\lambda}^{(g+1)}$ denotes the i-th optimal solution among λ offspring at generation g + 1. Each offspring is a perturbed trace, and the perturbation is randomly sampled from the Gaussian distribution. The classification accuracy of λ offspring is calculated separately and sorted in ascending order. $\delta_{1:\lambda}^{(g+1)}$ has the smallest fitness value, i.e., the lowest classification accuracy.

Algorithm 2 : Generating perturbation

```
Input: $X_w^f$–website traces correctly classified by f as w
       f–target classifier
       $n_{burst}$– the dimension of bursts sequence
       Φ– injection function
       $\theta$– classification accuracy threshold
       $I_t$– maximum number of iterations
       λ– number of offspring
       $\mu$– number of optimal solutions selected from offspringing
       $m^{(g)}$– mean value of the Gassuian distribution at generation g
       $\sigma^{(g)}$– step size at generation g
       $\mu_{eff}$–variance effective selection mass
       $c_1$– learning rate for the rank-one update of the covariance matrix
       $c_C$– learning rate for cumulation for the rank-one update of the covariance matrix
       $c_\sigma$–learing rate for cunulation for the step-size control
       $d_\sigma$– damping parameter for step-size update
       $p_C^{(g)}$– the evolution path of $C$ at generation g, initialize with zero vector
       $p_\sigma^{(g)}$– the evolution path of $\sigma$ at generation g, initialize with zero vector
```

(Continued)

Algorithm 2 (continued)

```
        $C^{(g)}$ –covariance matrix at generation g, initialize with Identity matrix
Output: $\delta_w$– perturbation for website w
1. $X_w^{f'} \leftarrow X_w^f$
2. $\delta_w$ ←initialize with zero vector
3. for iteration i in (1, 2, 3, ...,$I_t$) do
4.     for offspring j in (1, 2, 3, ...,$\lambda$) do
5.         $\delta[j] \leftarrow$ sample from N (m, $\sigma^2$C)
6.         $\delta[j] \leftarrow$ round ($\delta[j]$)
7.         fitness[j] $\leftarrow (f(\Phi\left(X_w^{f'}, \delta[j]\right) = f(X_w^f)))/\left|X_w^f\right|$
8.     end for
9.     $f_{sort} \leftarrow$ sort fitness and return index (ascending order)
10.    $f_{best}, \delta_{best} \leftarrow$ fitness $[f_{sort}[0]], \delta[f_{sort}[0]]$
11.    $\delta_w \leftarrow \delta_w + |\delta_{best}|$
12.    $X_w^{f'} \leftarrow \Phi\left(X_w^f, \delta_w\right)$
13.    if $f_{best} < \theta$ then
14.        return $\delta_w$
15.    end if
16.    update $m$, $p_\sigma$, $\sigma$, $p_C$, $C$ using Eqs. (4)–(7), (9), respectively
17. end for
18. return $\delta_w$
```

5 Experimental Setup

5.1 Datasets

Our experiments use publicly available datasets provided by Sirinam et al. [6] and Rimmer et al. [24], which are frequently used to evaluate WF attacks and defenses [14–16,18]. There are 95 websites in the Sirinam dataset, each with 1000 instances. Mockingbird [17] also uses the dataset provided by Sirinam but preprocesses it by removing instances with less than 50 packets or where the first packet was sent from the server, as the client should send the first packet to establish a connection with the server. Mockingbird then converts the preprocessed Sirinam dataset containing 95 websites and 512 traffic instances each website into burst sequences. For convenience, we straightforwardly use the dataset processed by Mockingbird. Rimmer provides four datasets: Rimmer100, Rimmer200, Rimmer500, and Rimmer900. Each has 100, 200, 500, and 900 websites and 2500 instances per website, respectively. Mockingbird splits traces into exclusive training and test sets with a ratio of 1:1 and uses 10% of the training set as a validation set. We split the Rimmer dataset into exclusive training, validation, and test sets with the recommended ratio of 9:0.5:0.5. Additionally, because Rimmer uses packet sequences as traffic feature, we need to convert them into burst sequences when generating and inserting perturbations. However, when fed to the model, we still use packet sequence as traffic feature.

5.2 Target Model

We evaluate our attack on three advanced DNN models: DF [6], VarCnn [5],and AWF [24]. These three models mainly consist of convolution and fully connected layers, but their complexities are very different. AWF with the simplest structure has the lowest accuracy and it is easiest to fool. DF has a higher accuracy than AWF because it has more fully connected layers for classification and more convolutional layers for feature extraction [13]. VarCnn is the most accurate, has the most intricate model structure, and is built on the ResNet-18 architecture. Despite reducing the number of parameters using dilated convolutions to reduce time and memory overheads, VarCnn's computational complexity is still 3.7 times higher than that of DF as shown in Table 3. The training parameters for DF and VarCnn are approximately 26 times those of AWF. Their computational complexities are 42 and 157 times that of AWF, respectively.

Table 3: The complexity of the target model

Model	Trainable Parameters	Complexity (FLOPS)
AWF	147k	11M
DF	3979k	463M
VarCnn	3893k	1728M

5.3 *Evaluation Metrics*

The defensive goal of this paper is to make the perturbed traces misclassified by the target model. One of the metrics used to evaluate the defensive capability is accuracy, which is the proportion of perturbed traces correctly classified out of the original samples correctly classified. For a given DNN model f and dataset X, accuracy is defined as:

$$Acc = \frac{1}{|X^f|} \sum\nolimits_{x \in X^f} [f(\varphi[x, \delta_x]) = f(x)] \tag{12}$$

where X^f denotes traces that are correctly classified by model f, and its size is strictly less than $|X|$. δ_x denotes perturbation injected into trace x, and all traces belonging to the same website w use the same perturbation $\delta_w(\delta_x = \delta_w, x \in w)$.

For WF defense, generating adversarial samples involves injecting dummy packets into the original traces. The link load will be too high if too many dummy packets are injected, affecting the user's experience. As a result, we also consider bandwidth overhead as a different metric for evaluating defense performance. The ratio of the total number of injected dummy packets to the total number of original packets is known as bandwidth overhead:

$$BWO = \frac{\sum x \in X^f \, \|\delta_x\|_1}{\sum x \in X^f \, \|x\|_1} \tag{13}$$

where $\|\ldots\|_1$ denotes the l_1 norm of the vector, i.e., the sum of the absolute values of all elements.

5.4 *CMAES-WFD Configuration*

Based on the trained DNN model, CMAES-WFD generates website-oriented perturbation. In order to find the optimal values of θ and γ, we selected several combinations of θ and γ and generated corresponding perturbations. After that, we looked at bandwidth overhead and accuracy to determine the best values for θ and γ. When using CMAES to solve the optimization problem, we initialized the parameters recommended in the literature. Table 4 displays the initialization values.

Table 4: Formulas for initialization of parameters

Parameters	Meanings	Initialization formulas
$\boldsymbol{\lambda}$	Number of offspring	$\lambda = 4 + floor\,(3 * logn_{burst})$
$\boldsymbol{\mu}$	Number of optimal solutions selected from the offspring	$\mu = floor\,(\lambda/2)$
$\boldsymbol{m}$	Mean value	Sample from a standard multivariate normal distribution
$\boldsymbol{\sigma}$	Step size	Initialized with 0.3
$\boldsymbol{weights}$	Recombination weights	$weights = \log(\mu + 1/2) - \log(1:\mu)$ $weights = weights/sum\,(weights)$
$\boldsymbol{\mu_{eff}}$	Variance effective selection mass	$\mu_{eff} = sum\,(weights)^2/sum\left(weights^2\right)$
$\boldsymbol{c_C}$	Cumulative learning rate for rank-one update of C	$c_C = \left(4 + \frac{\mu_{eff}}{n_{burst}}\right) \Big/ \left(n_{burst} + 4 + 2 * \frac{\mu_{eff}}{n_{burst}}\right)$
$\boldsymbol{c_1}$	Learning rate for rank-one update of C	$c_1 = 2/\left((n_{burst} + 1.3)^2 + \mu_{eff}\right)$
$\boldsymbol{c_\sigma}$	Cumulative learning rate of σ	$c_\sigma = \left(\mu_{eff} + 2\right) / \left(n_{burst} + \mu_{eff} + 5\right)$
$\boldsymbol{d_\sigma}$	Damping parameters of σ	$d_\sigma = 1 + 2 * \max\left(0, sqrt\left(\frac{\mu_{eff} - 1}{n_{burst} + 1}\right) - 1\right) + c_\sigma$
$\boldsymbol{p_C}$	Evolution path of C	Initialized with zero vectors
$\boldsymbol{p_\sigma}$	Evolution path of σ	Initialized with zero vectors
$\boldsymbol{C}$	Covariance matrix	Initialized with identity matrix

The experiment was run in Ubuntu operating system and Python3.9 environment, and the server hardware was configured with Intel Xeon Gold 5218 and NVIDIA GeForce GTX 3080 Ti. When the target model is AWF, the average time required to generate adversarial samples was about 40 s. When the target model is DF, the average time is about 1 min 27 s, when the target model is VarCnn, the average time is about 2 min 16 s.

6 Experimental Results

This section primarily presents numerous experiments that evaluate the effectiveness of CMAES-WFD. The experiments also compare CMAES-WFD with other advanced adversarial-based WF defenses, demonstrating that CMAES-WFD can effectively defend against DNN-based WF attacks with lower bandwidth overhead.

1. **Choice of parameters $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$.** The threshold θ and the number of optimal variances γ selection directly impact the accuracy and bandwidth overhead. To select the best θ and γ, we evaluated the accuracy and bandwidth overhead of CMAES-WFD against DF using various combinations of θ and γ on the Sirinam dataset. As θ and γ vary, the bandwidth overhead and accuracy trends are shown in Fig. 5. It is evident that as θ increases, the DF's accuracy gradually rises while bandwidth overhead gradually decreases when γ is fixed. This is because when θ

is small, the optimization condition is stricter, requiring more iterations and perturbations (injecting more dummy packets to satisfy the optimization condition) thus correspondingly higher bandwidth overhead. When θ is constant, as γ increases, the DF's accuracy gradually decreases while bandwidth overhead shows a trend of decreasing first and then increasing. When γ is small, fewer dummy packets are injected at each iteration, requiring more iterations to satisfy the optimization condition, resulting in higher bandwidth overhead. When γ is large, despite a decrease in the number of iterations, there is an increase in the number of dummy packets injected per iteration, which also has a significant bandwidth overhead.

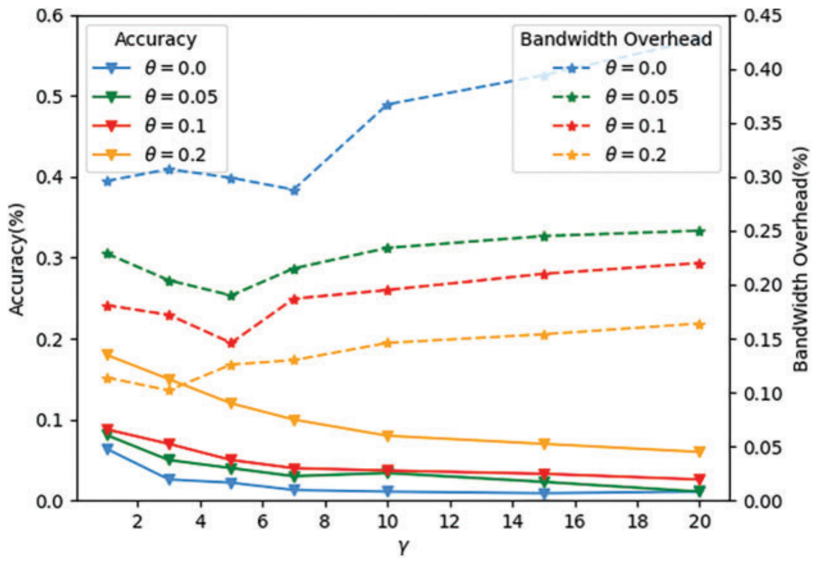


Figure 5: Accuracy and bandwidth overhead with various combinations of θ and γ (θ denotes the accuracy threshold of the objective function, and γ denotes the number of optimal variances)

We use a quantitative approach to calculate the F_1 score of different combinations of θ and γ to achieve a trade-off between accuracy and bandwidth overhead. The combination with the highest F_1 score was then chosen as the best values for θ and γ. Table 5 displays the F_1 scores. Usually, the F_1 score considers two metrics simultaneously, and the higher the value of both metrics, the higher the F_1 score. In this paper, we aimed for lower bandwidth overhead and lower accuracy. We developed a modified version of the F_1 score (Eq. (13)) to achieve this. Additionally, apply the maximum value normalization to the bandwidth overhead and accuracy when calculating the F_1 score. The F_1 score is highest at $\theta = 0.1$ and $\gamma = 5$, and bandwidth overhead and accuracy are 14.6% and 5.4%, respectively, at these values.

$$F_1 = 2 * \frac{\left(1 - \frac{Acc}{\max(Acc)}\right) * \left(1 - \frac{BWO}{max(BWO)}\right)}{\left(1 - \frac{Acc}{\max(Acc)}\right) + \left(1 - \frac{BWO}{max(BWO)}\right)} \tag{14}$$

2) **Generalization ability of parameters.** Although we quantified selected $\theta = 0.1$ and $\gamma = 5$ based on the F_1 score, we only tested it on one dataset and a single model. We evaluated the performance of $\theta = 0.1$ and $\gamma = 5$ on the Sirinam dataset against VarCnn and AWF to confirm their suitability for various models. We also selected the Rimmer500 and Rimmer900 datasets

and calculated the bandwidth overhead and accuracy against DF, VarCnn, and AWF to verify the performance of $\theta = 0.1$ and $\gamma = 5$ on various datasets.

Table 5: F_1 scores with various combinations of θ and γ

$\boldsymbol{\theta}$ (Accuracy threshold)	$\boldsymbol{\gamma}$ (The number of optimal variance)						
	1	3	5	7	10	15	20
0.0	0.416	0.422	0.449	0.482	0.246	0.143	1.896e-09
0.05	0.504	0.605	0.648	0.619	0.587	0.573	0.552
0.1	0.541	0.605	**0.680**	0.648	0.645	0.627	0.619
0.2	5.162e-09	0.264	0.477	0.523	0.596	0.632	0.640

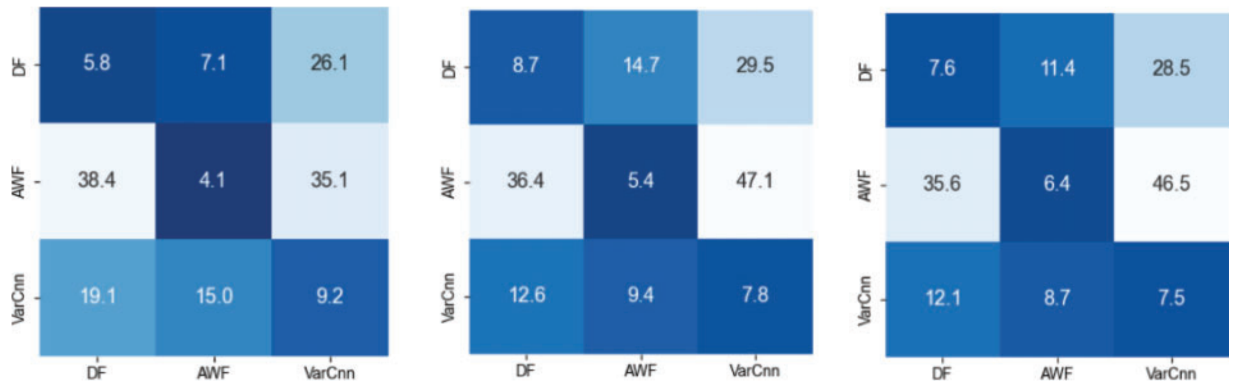
The results in Table 6 show that CMAES-WFD has the best defense performance against AWF model and the worst defense performance against VarCnn model. This is because VarCnn has better classification performance and more complex structure compared with AWF, so VarCnn is more difficult to fool than AWF. Overall, the accuracy of VarCnn could be reduced to below 8.1% with a bandwidth overhead of less than 20.5%, and the accuracy of DF could be reduced to below 8.3% with a bandwidth overhead of less than 14.6%. Meanwhile, the accuracy of AWF was less than 6.7%, with bandwidth overhead below 7.4%. The experimental results demonstrated that $\theta = 0.1$ and $\gamma = 5$ can be easily generalized to various models and datasets.

Table 6: Accuracy and bandwidth overhead on different models and datasets with $\theta = 0.1$ and $\gamma = 5$

Dataset	AWF		DF		VarCnn	
	Acc	BWO	Acc	BWO	Acc	BWO
Sirinam	4.8%	7.4%	5.4%	14.6%	8.1%	20.5%
Rimmer500	6.7%	5.1%	8.3%	10.8%	7.8%	18.3%
Rimmer900	5.6%	4.2%	6.4%	8.2%	6.7%	17.5%

3) **Transferability among different models:** Previous research [42–44] has demonstrated that adversarial samples have transferability, meaning that adversarial samples designed to attack a given classifier can also fool other classifiers. In this paper, the transferability of perturbations refers to the ability of perturbations generated for a target model to have defensive effects still when fed into other unknown models [13]. Due to the impracticality of generating perturbations for all possible models and the defender's inability to predict which classifier the attacker will use; this transferability feature is essential for WF defense. In order to evaluate the transferability of CMAES-WFD, we generated perturbations for each target model, then fed perturbed traces into other different models and calculated the accuracy.

The experiments still used AWF, DF, and VarCnn as target models. The transferability of perturbation generated by CMAES-WFD among the three models is shown in Fig. 6. The rows indicate target models generating perturbations, and the columns indicate test models. The perturbations computed by three target models have certain transferability, among which perturbations against VarCnn have the largest transfer rate among the three models. For example, on Rimmer900, the accuracy of AWF and DF for perturbed traces generated based on VarCnn is 12.1% and 7.5%, respectively. In contrast, perturbations generated against AWF had the lowest transfer rate, with DF and VarCnn achieving 35.6% and 46.5% accuracy for perturbed traces, respectively. Since AWF has the simplest structure and is the easiest to be fooled, the results were to be expected. Because the higher the accuracy of the model and the more complex the model, the stronger the deception ability of the adversarial sample generated for the model, so that it can fool other target models with lower complexity. Therefore, VarCnn has the highest transferability rate and the model with higher complexity can be used as the target model for searching the universal perturbation. As demonstrated by the experiments, we conclude that VarCnn, with its most complex structure, outperforms other DNN models regarding in resisting adversarial perturbations and generating transferable perturbations.



(a) Sirinam (b) Rimmer500 (c) Rimmer900

Figure 6: Transferability of CMAES-WFD perturbations among models

4) **Adversarial training:** Adversarial training refers to adding adversarial samples to the training set and retraining the model [38]. The literature [17,45] has pointed out that training models on adversarial samples is the most effective defense against adversarial attacks, and an attack model with adversarial training can identify the true class of adversarial samples to a large extent. We assume that the attacker can access the CMAES-WFD source code and generates perturbations for adversarial training.

We generated perturbations using the same parameters, and then retrained AWF, DF and VarCnn using the perturbed traces as the training set. As shown in Table 7, the accuracy of AWF, DF and VarCnn with adversarial training are 52.4%, 56.7%, 62.8%, respectively. This shows that even if the attacker can fully reproduce CMAES-WFD and use the generated adversarial samples to train the model, their model's accuracy is only about 60%, which greatly reduces the classification performance of the model, indicating that perturbations generated by our method still had a defensive impact on AWF, DF and VarCnn with adversarial training. The proposed CMAES-WFD generates perturbations by randomly sampling from the Gaussian distribution, which has a certain degree of randomness. Therefore, even if the target DNN-based model is trained with perturbed traces, CMAES-WFD still has a certain defensive ability.

5) **Effects of monitored website number:** In practical scenarios, researchers may focus on different monitored websites. Thus, monitored website numbers may also vary greatly. This experiment further explores the effect of the number of monitored websites on the defensive performance of CMAES-WFD. The experiment used four Rimmer datasets, Rimmer100, Rimmer200,

Rimmer500, and Rimmer900, with DF as the target model. The results are depicted in Fig. 7. According to the experimental results, DF's accuracy on perturbed traces decreased with lower bandwidth overhead as the number of monitored websites increased when using the same parameters. For instance, with only 8.2% bandwidth overhead, DF's accuracy on perturbed traces on Rimmer900 was 7.2%. The accuracy of DF on Rimmer100 was 9.1%, which was similar to that on Rimmer900. However, the bandwidth overhead was much higher on Rimmer100, at 18.3%, more than twice that on Rimmer900. This is due to the limited representation ability of DNN in the feature space. The more monitored websites are, the less distinguishable among different websites will become in the DNN feature space, making it easier to generate perturbations.

Table 7: Accuracy of models with adversarial training

Model	Accuracy
AWF	52.4%
DF	56.7%
VarCnn	62.8%

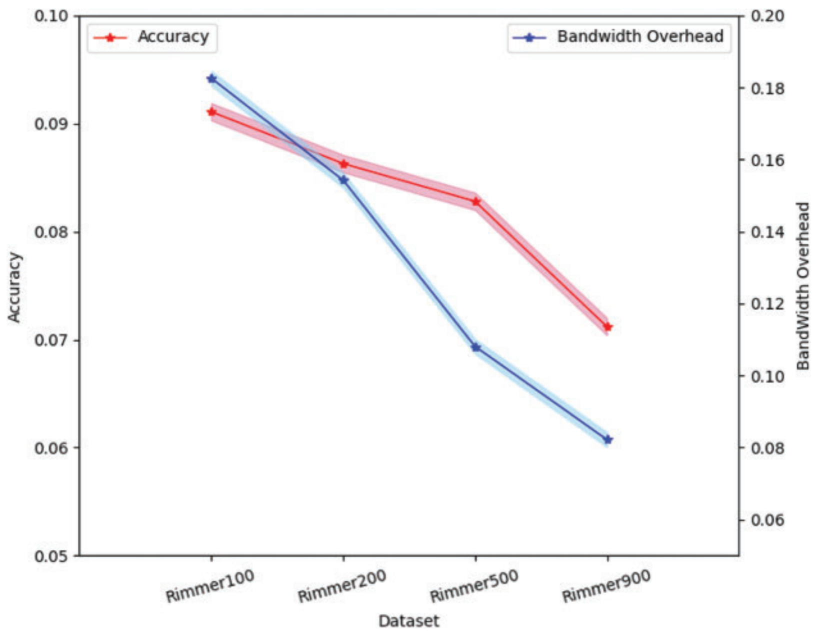
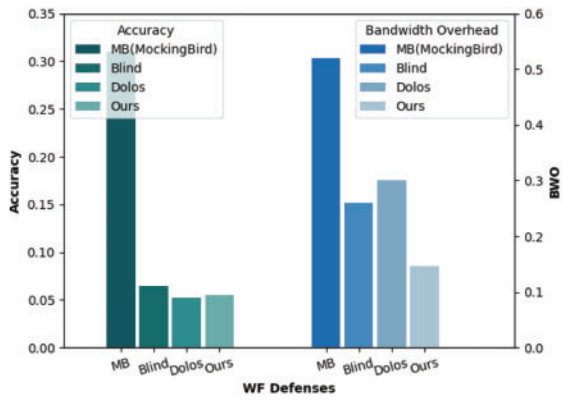


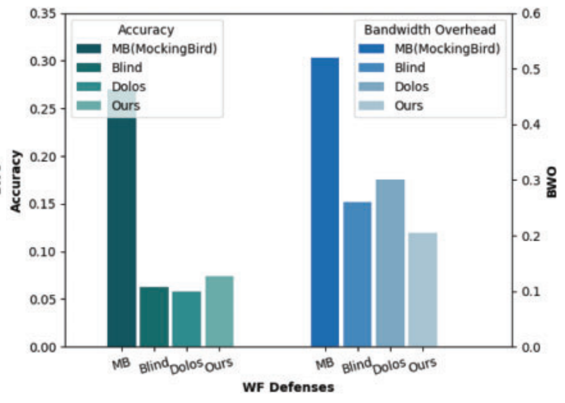
Figure 7: Accuracy and bandwidth overhead with different monitored websites numbers

6) **Comparison:** We compare CMAES-WFD with three representative adversarial-based WF defenses, and the results are shown in Fig. 8. We evaluate the performance of these WF defenses against DF and VarCnn on Sirinam and Rimmer900 datasets, respectively. As can be seen, Mockingbird [17] reduced accuracy to around 30% but still failed to provide sufficient effective protection, and the overhead of Mockingbird was over 50%, which may hurt user experience. Dolos [15] reduces accuracy to around 5% with 30% bandwidth overhead. However, Dolos is trace-oriented when generating adversarial patches and inserts adversarial patches at specific

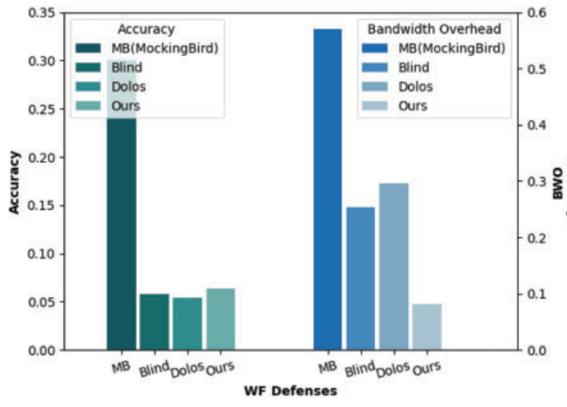
locations. This is unsuitable for other traces of the same website, resulting in high bandwidth overhead. Dolos is better than us at reducing accuracy because we consider the balance between accuracy and bandwidth overhead, and our bandwidth overhead is lower, so the accuracy is higher than Dolos. Blind [18] reduces accuracy to less than 6% with 25% bandwidth overhead. Although it achieves real-time traffic defense, it has higher bandwidth overhead and requires computing the gradient of the loss function, which is unrealistic in a black-box scenario. The comparison shows that CMAES-WFD achieves a good balance between accuracy and bandwidth overhead by reducing the classification accuracy of both target models to below 8.2% and having a significantly lower bandwidth overhead than other WF defenses. This opens up the possibility of real-world deployment, because excessive bandwidth overhead can seriously affect the user experience, after all, users do not want too much network latency when browsing websites.



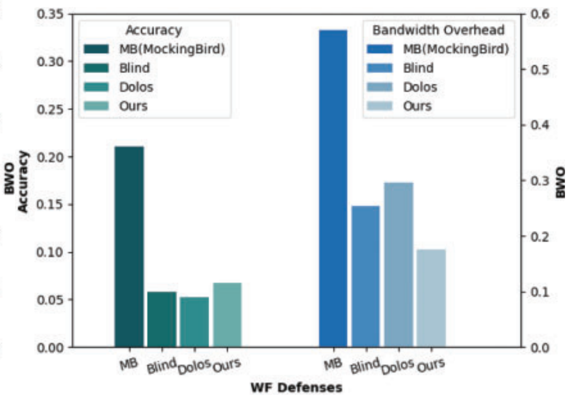
(a) Sirinam dataset, DF model



(b) Sirinam dataset, VarCnn model



(c) Rimmer900 dataset, DF model



(d) Rimmer900 dataset, VarCnn model

Figure 8: Comparison of CMAES-WFD with adversarial-based WF defenses (lower is better). (a) The comparison of accuracy and bandwidth overhead of CMAES-WFD, MockingBird, Dolos and Blind against DF model on Sirinam dataset; (b) The comparison of accuracy and bandwidth overhead of CMAES-WFD, MockingBird, Dolos and Blind against VarCnn model on Sirinam dataset; (c) The comparison of accuracy and bandwidth overhead of CMAES-WFD, MockingBird, Dolos and Blind against DF model on Rimmer900 dataset; (d) The comparison of accuracy and bandwidth overhead of CMAES-WFD, MockingBird, Dolos and Blind against VarCnn model on Rimmer900 dataset

7 Limitations and Future Work

When generating perturbations, CMAES-WFD requires prior knowledge of the target website that the user intends to access. This assumption is common in WF defenses [14,15,29], which focus on generating perturbations based on specific traffic patterns. However, website traces may significantly differ due to geographical location variations. Consequently, some WF defenses aim to generate universal adversarial perturbations, but these attempts increase bandwidth overhead. Future work could focus more on universal adversarial perturbations, reducing dependence on specific traces and using lower bandwidth overhead.

In addition, CMAES-WFD is designed to defend against WF attacks that use packet sequences as features and are based on DNN models. Therefore, our defense may not apply to non-DNN models using manually extracted features such as packet numbers and intervals. Future work can explore the combination of our defense with defense against non-DNN techniques. For example, non-DNN models usually use timing features, which can increase some delay to the original packets or dummy packets, so that the generated adversarial samples are not only applicable to DNN models, but also have a certain deception effect on non-DNN models.

Finally, because CMAES-WFD is based on burst sequence when computing adversarial samples, that is, a complete traffic sequence is required, it cannot be applied to real-time traffic yet. In the subsequent research, we will focus on how to improve the algorithm to apply to real-time traffic.

8 Conclusion

A WF defense based on adversarial samples that provides effective protection against DNN-based WF attacks with lower bandwidth overhead is proposed in this paper as CMAES-WFD. CMAES-WFD only requires black-box feedback from the target model to generate adversarial perturbations for various websites by solving specific optimization problems. Additionally, to reduce bandwidth overhead, we only inject dummy packets at locations with higher variances, following the updated principle of increasing variance along the successful search direction in CMAES.

Experiments show that our method outperforms the majority of previous defenses by reducing the accuracy of the target models to less than 8.3% with a bandwidth overhead of no more than 20.5%. We chose three advanced DNN-based models as our target models. In addition, we demonstrate that the adversarial perturbation vectors generated against VarCnn have the best transferability among the three models and can defeat unknown attack models. Finally, we also demonstrate that CMAES-WFD is somewhat resistant to adversarial training.

Acknowledgement: The authors would like to express their gratitude to EditSprings (https://www.editsprings.cn) for the expert linguistic services provided.

Funding Statement: This work was supported by the Key JCJQ Program of China: 2020-JCJQ-ZD-021-00 and 2020-JCJQ-ZD-024-12.

Author Contributions: The authors confirm contribution to the paper as follows: Study conception and design: Di Wang, Jinlong Fei; data collection: Di Wang; analysis and interpretation of results: Di Wang, Jinlong Fei, Yuefei Zhu, Maohua Guo; draft manuscript preparation: Di Wang, Jinlong Fei, Yuefei Zhu, Maohua Guo. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data and materials used to support the findings of this study are available from the corresponding author upon request after acceptance.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] T. A. Ghaleb, "Wireless/website traffic analysis & fingerprinting: A survey of attacking techniques and countermeasures," in *Int. Conf. Cloud Comput.*, New York, NY, USA, 2015, pp. 1–7.
[2] A. Panchenko *et al.*, "Website fingerprinting at internet scale," in *Proc. Netw: Distrib. Syst. Secur. Symp.*, Reston, VA, USA, 2016, pp. 1–15.
[3] J. Hayes and G. Danezis, "*k*-fingerprinting: A robust scalable website fingerprinting technique," in *USENIX Secur. Symp.*, Austin, TX, USA, 2016, pp. 1187–1203.
[4] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *23rd USENIX Secur. Symp. (USENIX Secur. 14)*, San Diego, CA, USA, 2014, pp. 143–157.
[5] S. Bhat, D. Lu, A. Kwon, and S. Devadas, "Var-CNN: A data-efficient website fingerprinting attack based on deep learning," in *Proc. Priv. Enhanc. Technol.*, vol. 2019, no. 4, pp. 292–310, 2019. doi: 10.2478/popets-2019-0070.
[6] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proc. 2018 ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, 2018, pp. 1928–1943.
[7] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an efficient website fingerprinting defense," in *Comput. Secur.–ESORICS 2016: 21st Eur. Symp. Res. Comput. Secur.*, Heraklion, Greece, Sep. 26–30, 2016, pp. 27–46.
[8] Y. Dong *et al.*, "Boosting adversarial attacks with momentum," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, Utah, USA, 2018, pp. 9185–9193.
[9] S. M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, Nevada, USA, 2016, pp. 2574–2582.
[10] W. D. L. Cadena, A. Mitseva, J. Hiller, J. Pennekamp, and A. Panchenko, "TrafficSliver: Fighting website fingerprinting attacks with traffic splitting," in *CCS '20: 2020 ACM SIGSAC Conf. Comput. Commun. Secur.*, USA, 2020, pp. 1971–1985.
[11] R. Tang, G. Shen, C. Guo, and Y. Cui, "SAD: Website fingerprinting defense based on adversarial examples," in *Int. Conf. Secur. Priv. New Comput. Environ.*, Qinhuangdao, China, 2021, pp. 88–102.
[12] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *Proc. 2nd Int. Conf. Learn. Rep.*, Banff, Canada, 2014, pp. 1–10.
[13] D. Li, Y. Zhu, M. Chen, and J. Wang, "Minipatch: Undermining DNN-based website fingerprinting with adversarial patches," *IEEE Trans. Inf. Forensics Secur.*, vol. 17, no. 8, pp. 2437–2451, 2022. doi: 10.1109/TIFS.2022.3186743.
[14] A. M. Sadeghzadeh, B. Tajali, and R. Jalili, "AWA: Adversarial website adaptation," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 3109–3122, 2021. doi: 10.1109/TIFS.2021.3074295.
[15] S. Shan, A. N. Bhagoji, H. Zheng, and B. Y. Zhao, "Patch-based defenses against web fingerprinting attacks," in *Proc. 14th ACM Workshop Artif. Intell. Secur.*, Korea, 2021, pp. 97–109.
[16] C. Hou, G. Gou, J. Shi, P. Fu, and G. Xiong, "WF-GAN: Fighting back against website fingerprinting attack using adversarial learning," in *2020 IEEE Symp. Comput. Commun. (ISCC)*, Rennes, France, 2020, pp. 1–7.

[17] M. S. Rahman, M. Imani, N. Mathews, and M. Wright, "Mockingbird: Defending against deep-learning-based website fingerprinting attacks with adversarial traces," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, no. 6, pp. 1594–1609, 2020. doi: 10.1109/TIFS.2020.3039691.

[18] M. Nasr, A. Bahramali, and A. Houmansadr, "Defeating DNN-based traffic analysis systems in real-time with blind adversarial perturbations," in *USENIX Secur. Symp.*, Vancouver, B.C., Canada, 2021, pp. 2705–2722.

[19] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier," in *Proc. 2009 ACM Workshop Cloud Comput. Secur.*, Chicago, IL, USA, 2009, pp. 31–42.

[20] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proc. 10th Annu. ACM Workshop Priv. Electron. Soc.*, Chicago, IL, USA, 2011, pp. 103–114.

[21] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from adistance: Website fingerprinting attacks and defenses," in *Proc. ACM Conf. Comput. Commun. Secur.*, Raleigh, NC, USA, 2012, pp. 605–616.

[22] T. Wang and I. Goldberg, "Improved website fingerprinting on tor," in *Proc.12th ACM Wrkshop Priv. Electron. Soc.*, Berlin, Germany, 2013, pp. 201–212.

[23] K. Abe and S. Goto, "Fingerprinting attack on Tor anonymity using deep learning," in *Proc. Asia Pac. Adv. Netw.*, vol. 42, no. 1, pp. 15–20, 2016.

[24] V. Rimmer, D. Preuveneers, M. Juarez, T. V. Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," in *Proc. Netw., Distrib. Syst. Secur.: Symp.*, Reston, VA, USA, 2018, pp. 1–15.

[25] B. Sun, W. Yang, M. Yan, Y. Zhu, and Z. Bai, "A practical website fingerprinting defense approach with universal adversarial perturbations," in *2022 7th Int. Conf. Comput. Commun. Syst. (ICCCS)*, Wuhan, China, 2022, pp. 752–760.

[26] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-Boo, I still see you: Why efficient traffic analysis countermeasures fail," in *2012 IEEE Symp. Secur. Priv.*, San Francisco, CA, USA, 2012, pp. 332–346.

[27] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, "A systematic approach to developing and evaluating website fingerprinting defenses," in *Proc. 2014 ACM SIGSAC Conf. Comput. Commun. Secur.*, Scottsdale, AZ, USA, 2014, pp. 227–238.

[28] X. Cai, R. Nithyanand, and R. Johnson, "CS-BuFLO: A congestion sensitive website fingerprinting defense," in *Proc. 13th Workshop Priv. Electron. Soc.*, Scottsdale, AZ, USA, 2014, pp. 121–130.

[29] T. Wang and I. Goldberg, "Walkie-Talkie: An efficient defense against passive website fingerprinting attacks," in *USENIX Secur. Symp.*, Vancouver, BC, Canada, 2017, pp. 1375–1390.

[30] A. Abusnaina, R. Jang, A. Khormali, D. Nyang, and D. Mohaisen, "DFD: Adversarial learning-based approach to defend againstwebsite fingerprinting," in *Proc. IEEE INFOCOM2020-IEEE Conf. Comput. Commun.*, Toronto, ON, Canada, 2020, pp. 2459–2468.

[31] Y. Dong *et al.*, "Boosting adversarial altacks with momentum," in *Proc. IEEE/CVF Conf. Comput. Vs. Pattern Recognit.*, Salt Lake City, Utah, USA, 2018, pp. 9185–9193.

[32] C. Xiao, B. Li, J. Y. Zhu, W. He, M. Liu and D. Song, "Generatingadversarial examples with adversarial networks," in *Proc. 27th Int. Joint. Conf. Artif. Intell.*, Stockholm, Sweden, 2018, pp. 3905–3911.

[33] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, 2019. doi: 10.1109/TNNLS.2018.2886017.

[34] S. Henri, G. García, P. Serrano, A. Banchs, and P. Thiran, "Protecting against website fingerprinting with multihoming," in *Proc. Priv. Enhanc. Technol.*, vol. 2020, no. 2, pp. 89–110, 2020. doi: 10.2478/popets-2020-0019.

[35] J. Gong, W. Zhang, C. Zhang, and T. Wang, "Surakav: Generating realistic traces for a strong website fingerprinting defense," in *2022 IEEE Symp. Secur. Priv. (SP)*, San Francisco, CA, USA, 2022, pp. 1558–1573.

[36] J. Gong and T. Wang, "Zero-delay lightweight defenses against website fingerprinting," in *Proc. 29th USENIX Conf. Secur. Symp.*, Berkeley, CA, USA, 2020, pp. 717–734.

[37] M. S. Rahman, P. Sirinam, N. Atthews, K. G. Gangadhara, and M. Wright, “Tik-Tok: The utility of packet timing in website finger-printing attacks,” in *Proc. Priv. Enhanc. Technol.*, vol. 2020, no. 1, pp. 5–24, 2020. doi: 10.2478/popets-2020-0043.

[38] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in *Proc. 3rd Int. Conf. Learn. Rep.*, San Diego, CA, USA, 2015, pp. 1–11.

[39] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik and A. Swami, “The limitations of deep learning in adversarial settings,” in *2016 IEEE Eur. Symp. Secur. Priv. (Euro S&P)*, Saarbrücken, Germany, 2016, pp. 372–387.

[40] S. M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, “Universal adversarial perturbations,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, 2017, pp. 1765–1773.

[41] N. Hansen *et al.*, “The CMA evolution strategy: A tutorial,” *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, 2016. doi: 10.1162/106365601750190398.

[42] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in *2017 IEEE Symp. Secur. Priv. (SP)*, San Jose, CA, USA, 2017, pp. 39–57.

[43] A. Demontis *et al.*, “Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks,” in *28th USENIX Secur. Symp. (USENIX Secur. 19)*, Santa Clara, CA, USA, 2019, pp. 321–338.

[44] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, “A critical evaluation of website fingerprinting attacks,” in *Proc. 2014 ACM SIGSAC Conf. Comput. Commun. Secur.*, Scottsdale, AZ, USA, 2014, pp. 263–274.

[45] X. Zhang, J. Hamm, M. K. Reiter, and Y. Zhang, “Statistical privacy forstreaming traffic,” in *26th Annu. Net. Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, CA, USA, 2019, pp. 24–27.