WFGUARD: an Effective Fuzzing-testing-based Traffic Morphing Defense against Website Fingerprinting

Zhen Ling[y], Gui Xiao[z], Lan Luo[x] , Rong Wang[y], Xiangyu Xu[y], and Guangchi Liu[y]
[y]School of Computer Science and Engineering, Southeast University, China
[z]School of Cyber Science and Engineering, Southeast University, China
Email: {zhenling, xiaogui, junowang, xy-xu, gc-liu}@seu.edu.cn
[x]School of Computer Science and Technology, Anhui University of Technology, China
Email: lluo@ahut.edu.cn

***Abstract*—Website fingerprinting (WF) attack is a type of traffic analysis attack. It enables a local and passive eavesdropper situated between the Tor client and the Tor entry node to deduce which websites the client is visiting. Currently, deep learning (DL) based WF attacks have overcome a number of proposed WF defenses, demonstrating superior performance compared to traditional machine learning (ML) based WF attacks. To mitigate this threat, we present WFGUARD, a fuzzing-testing-based traffic morphing WF defense technique. WFGUARD employs fine-grained neuron information within WF classifiers to design a joint optimization function and then applies gradient ascent to maximize both neurons value and misclassification possibility in DL-based WF classifiers. During each traffic mutation cycle, we propose a gradient based dummy traffic injection pattern generation approach, continuously mutating the traffic until a pattern emerges that can successfully deceive the classifier. Finally, the pattern present in successful variant traces are extracted and applied as defense strategies to Tor traffic. Extensive evaluations reveal that WFGUARD can effectively decrease the accuracy of DL-based WF classifiers (e.g., DF and Var-CNN) to a mere 4.43%, while only incurring an 11.04% bandwidth overhead. This highlights the potential efficacy of our approach in mitigating WF attacks.**

***Index Terms*—Anonymous communication systems, Website fingerprinting, Fuzzing testing**

I. INTRODUCTION

Tor is one of the most widely used anonymous communication systems due to its outstanding anonymity protection capability. According to Tor Metrics [32], there are about three million active users utilize Tor to protect their privacy. However, owing to its popularity, it attracts lots of researchers to de-anonymize users' privacy using various traffic analysis techniques [18]–[20]. Website fingerprinting (WF) attack is a traffic analysis attack that enables a local and passive eavesdropper between the Tor client and the Tor entry node to infer which websites the client is visiting. Figure 1 shows the attack model. The local WF attackers (e.g., an Internet service provider (ISP), or a local network administrator) passively record and collect the Tor network traffic without modifying, delaying or decrypting any packet of traces.

The WF attack can be modeled as a supervised classification problem, in which the traces of each website are labeled and used to train various WF classifiers. Traditional machine learning (ML) based WF attacks, such as k-NN [33], CUMUL [23] and k-FP [12], are studied to achieve about 90% accuracy. However, the performance of ML-based attacks rely on the selection of hand-crafted traffic features, such as traffic burst, packet timing interval, packet direction and so on. To resolve the problems, a large number of efficient deep learning (DL) based WF attacks [1], [3], [27], [28], [30], [31] are proposed to automatically extract the traffic features without any feature engineering. What is more, the DL-based WF attacks show better performance than ML-based WF attacks.

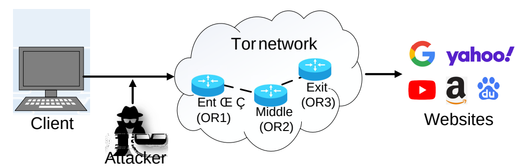

Fig. 1. Website fingerprinting attack

To mitigate the WF attacks, various WF defenses are introduced to protect users privacy, such as BuFLO [8], WTF-PAD [16], Walkie-Talkie [35], FRONT [9], Mockingbird [26], Surakav [10] and so on. These defenses perturb the traffic by injecting dummy packets and/or delaying packets so as to eliminate the distinguishable traffic features of each website and confuse the WF attackers.

However, the sophisticated architectures of DL-based WF attacks pose challenges when designing effective defenses to eliminate high-level traffic features. Consequently, many existing defenses show poor performance against DL-based WF attacks with unacceptably high bandwidth and/or latency overhead. To address the issues, the design of more efficient defense methods that effectively deceive DL-based WF classifiers, leading to misclassifications on Tor traffic, becomes necessary. One potential solution is to leverage essential information of deep learning network to design defense methods. Additionally, we notice that fuzzing testing deep neural network (DNN) shows promise in improving WF defenses. Fuzzing testing DNN aims to generate various inputs to trigger the decision logic of DNN and identify a large number of unexpected behaviors, such as misclassification. This aligns with the goals of WF defenses. Nevertheless, none

* Corresponding author: Dr. Lan Luo of Anhui University of Technology, China.

of the existing defenses combine fine-grained information of deep learning network with fuzzing testing DNN technique to create effective and efficient defenses against DL-based WF classifiers.

In this paper, we employ fuzzing testing DNN technique to design a fuzzing-testing-based traffic morphing defense (WFGuard) against WF attacks. The accuracy of DL-based WF classifiers is determined by their neurons values. These values, susceptible to differences due to unique traffic features inherent to specific websites, play a crucial role in guiding the decision-making process of the classifier. Thus, we can leverage these neurons values to construct a fuzzing objective function and influence them by continuously mutating traffic patterns. Such traffic manipulation can potentially change the classifier's decision logic, leading to possible misclassifications, thereby rendering the approach as a potent WF defense strategy. Specifically, we first elaborately select representative traffic traces for each website and initiate a seed pool. Next, we design a joint optimization function based on deliberately selected neurons valuess and misclassifications of WF classifiers. For each trace from the pool, gradient ascent is utilized to maximize the optimization function. The computed gradient is then processed with trace mutation strategies so as to obtain the injection pattern (i.e., the injection position and direction) for the trace. In this way, the joint optimization function guides the mutation on all traces of each website. In addition, we propose two mutation strategies to generate various injection patterns for comparison. We choose the more superior injection patterns, which can successfully deceive the attacker with the minimum bandwidth overhead and minimum accuracy of DL-based WF classifiers. Extensive evaluation shows that WFGuard reliably decreases the accuracy of DL-based WF classifiers (e.g., DF [31], Var-CNN [3]) to 4.43% with only 11.04% bandwidth overhead in the closed-world scenario and 6% with only 11.22% bandwidth overhead in the open-world scenario. It demonstrates that WFGuard outperforms the existing defense approaches [16], [22], [26].

In summary, our major contributions are as follows:

- To the best of our knowledge, we are the first to employ the fuzzing testing DNN technique to design WFGuard method against DL-based WF attackers. We can find one injection pattern for each website which minimize the DL-based WF classification accuracy. We utilize the fine-grained neuron information to design a joint optimization function which incorporates two parts: maximizing the neurons values and the number of the misclassification behaviors of DL-based WF classifiers.
- We leverage the gradient ascent method to maximize the joint optimization function. In particular, the dimension of gradient vector is the same as the input trace. Therefore, we obtain the injection positions and directions of dummy cell according to the index and sign of the gradient vector.
- We evaluate the feasibility and efficiency of WFGuard against DL-based WF classifiers through extensive experiments including a series of mutation strategies employed to generate various injection patterns for original traces. The efficiency of the injection patterns are evaluated with 800 traces of each website. The experimental results demonstrate that WFGuard can significantly decrease the accuracy of the DL-based WF classifiers to around 4.43% by only introducing less than 11.04% bandwidth overhead, and can effectively defend against DL-based traffic analysis attacks to preserve the communication privacy.

The rest of this paper is organized as follows. In Section II, we give the background of WF attacks, WF defenses and fuzzing testing deep neural networks which inspired us to design the WFGuard method. We introduce the threat model, basic idea, motivation and the details of WFGuard design in Section III. Then we conduct extensive experiments to evaluate the performance of WFGuard method as well as the existing WF defenses in Section IV. We review related work in Section V and conclude this paper in Section VI.

II. Background

This section covers necessary background. We briefly introduce the WF attack and defense techniques as well as the fuzzing testing deep neural network.

A. Website Fingerprinting Attack

WF attacks aim to undermine anonymity protection in anonymous communication systems, typically where users employ networks like Tor for browsing. Attackers passively gather raw network traffic between the Tor client and entry node, as depicted in Figure 1. They extract traffic features to form website fingerprints, using these to train an offline classifier. This classifier is then deployed at runtime to identify the specific websites visited by the potential victim.

Existing WF attacks fall into two categories: ML-based WF attacks [6], [12], [23], [24], [33], [34], [38] and DL-based WF attacks [1], [3], [27], [28], [31]. The ML-based WF attacks rely on expert knowledge to extract hand-crafted traffic features, such as traffic burst, packet timing interval, and packet direction, to train the classifier and leverage the output scores from the ML-based WF classifiers to infer the visited websites. However, the efficacy of ML-based WF classifiers heavily depends on the feature engineering.

To tackle this issue, since 2016, DL-based WF attacks have been introduced to leverage deep learning models to automatically extract high-dimensional traffic features to train the classifier for identifying different websites. Before the training process, raw traces are preprocessed to extract the Tor cells using the method proposed by [34]. Moreover, the preprocessed trace is padded into a fixed length for classifier input. This results in a trace sequence comprising +1, -1, and 0. +1 represents a Tor cell emitted from the Tor client to the website, -1 signifies a Tor cell sent in the opposite direction and 0 pads the trace to the fixed length.

B. Website Fingerprinting Defense

To protect user communication privacy, a series of WF defenses have been proposed. These defenses aim to hide the patterns of the traffic and ensure the anonymity of user communications. While existing defenses involve either injecting dummy packets or delaying real packets, both methods have their trade-offs. Injecting dummy packets alters the traffic patterns by introducing additional packets. However, this approach results in extra bandwidth overhead. On the other hand, delaying real data packets has a significant impact on the arrival time of the packets, which leads to additional latency overhead. This delay reduces the loading speed of websites and directly affects the overall browsing experience of users. As a result, the central objective in designing WF defenses is to strike a balance between the necessary overhead incurred and the overall effectiveness.

The existing defenses can be divided into two categories: feature-suppression-based WF defenses [4], [5], [7], [8], [13], [16], [21], [33], [35] and feature-morphing-based WF defenses [2], [9], [14], [17], [26], [29]. The feature-suppression-based WF defenses involve using traffic obfuscation methods to homogenize the traffic features of all websites. This approach aims to prevent the classifier from accurately classifying the websites. However, it takes a large overhead to achieve homogenizing the traffic features of all websites. Therefore, researchers propose the feature-morphing-based WF defenses, which aim to reshape the source traffic feature of current website into a different website by injecting dummy cells. These methods can successfully mislead the state-of-the-art DL-based WF classifiers with lower overhead.

C. Fuzzing Testing Deep Neural Networks

In traditional software testing domain, fuzzing testing is leveraged to detect huge amount of software vulnerabilities. The key idea of fuzzing testing is to generate random inputs to detect lots of incorrect software behaviors and potential flaws. The idea can be also employed for improving robustness of DNN. Therefore, fuzzing testing is also used to explore the decision boundaries of DNN and get more undesired behaviors, such as misclassification. In fuzzing testing DNN domain, existing methods [11], [15], [25], [36], [37] concentrate on generating various inputs by using different techniques to maximize neuron coverage and detect incorrect behaviors at the same time. Neuron coverage is a ratio of the number of unique activated neurons for all test inputs to the total number of neurons in the DNN. A neuron is activated if its output value is higher than a threshold value (e.g., 0). As we can see from the results of these methods, fuzzing testing DNN is able to maximize both the number of observed differential behaviors and the neuron coverage, which inspired us to apply the technique of fuzzing testing DNN to WF defense.

III. Fuzzing-testing-based Traffic Morphing Technique

In this section, we first introduce the threat model and motivation of our defense. Then, we present the basic idea of our fuzzing-testing-based traffic morphing technique. Finally, we elaborate on the critical design of our method step by step.

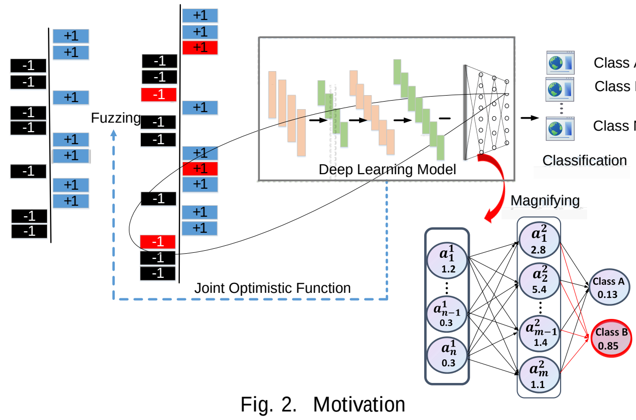

Fig. 2. Motivation

A. Threat model

The threat model of WF attacks is depicted in Figure 1. We assume a local and passive attacker is capable of identifying the individual website visited by a Tor user. A "local" attacker is one positioned somewhere between the Tor client and the Tor entry node. The term "passive" denotes that an attacker is able to observe and record Tor network traffic without the capacity to modify, delay, or drop packets. Such potential attackers include Internet Service Providers (ISP), Autonomous Systems (AS) and local network administrators that are positioned between the Tor client and the entry node. An attacker can collect labeled traffic and preprocess the traffic using the method proposed by [34] to train an offline classifier. Additionally, since the most sophisticated WF classifiers based on the traffic preprocessed method [34] include DF [31] and Var-CNN [3], we assume that the attacker deploys such WF classifiers to inspect the traffic and performs the WF attacks at runtime.

B. Motivation

The WF defense strategy is designed to create deceptive traffic patterns by inserting dummy cells. Its primary goal is to confuse DL-based WF classifiers while keeping overhead at a minimum. Drawing inspiration from the fuzzing testing approach, we utilize neuron information derived from DL-based classifiers as feedback for the fuzzing process. This feedback guides the mutation process, allowing us to identify an optimal injecting pattern with the fewest possible injections required to mislead the classifier.

It is crucial to highlight that the accuracy of DL-based WF classifiers is susceptible to the neurons values present within them. Due to the varying traffic features across different websites, there are disparities in the neurons values within the classifier. The decision-making mechanism of the classifier heavily depends on these neurons values and the weights among interconnected neurons. In light of this, we can leverage the neurons values as fuzzing feedback by employing mutation to the injection patterns. By doing so, we indirectly alter specific neurons values, resulting in changes to the classifier's decision logic. This process has the potential to cause misclassifications, making it an effective strategy for the WF defense.

representative traces act as the seeds for subsequent seed mutation.

We take a seed pool of a website as an example to illustrate the following steps. After constructing the initial seed pool of the website, we retrieve a seed x from the pool without repetition for trace mutation. For each seed in the seed pool, an upper bound of the number of mutations is used to control bandwidth overhead, namely the ratio of the number of the injected dummy cells to the number of the cells in the original trace. Denote the number of maximum mutations as M_α, to be the product of the number of cells in the original trace and a predefined coefficient α. In this way, we can effectively restrict the bandwidth overhead from exceeding α. The optimal value for α is determined through empirical experiments discussed in Section IV.

E. Joint Optimization Function

For the trace retrieved from the seed pool, we use a joint optimization function to guide effective mutations on the trace. We utilize the neuron information of WF classifier to design joint optimization function. The joint optimization function includes maximizing the values of selected activated neurons and misclassifications. If a mutated input is found to increase the values of activated neurons, it can cause the classifier to identify the input as a wrong label [11], [25]. Hence, we design a joint optimization function to assist WFGUARD to find the effective injection pattern. The function, also referred to as the objective function, is defined as

$$obj(x) = \lambda_1 \sum_{i}^{m} n_i(x) + \lambda_2 (c_1(x) - c_0(x)) \tag{1}$$

where $n_i(x)$ is the value of a selected neuron of which the value is intended to be increased, and m is the number of the selected neurons. The optimal value for m is determined through empirical experiments discussed in Section IV.

Since increasing the values of neurons during mutation can mislead the classifier, the first part of the expression $\sum_i^m n_i(x)$ in Equation (1) is designed in an attempt to maximize the sum values of all pre-selected neurons so as to cause the misclassifications. In order to maximize the neurons values, we propose two heuristic neuron selection strategies based on the activation count for each neuron. The activation count (denoted as C) for each neuron is obtained in advance by feeding p traces of each website into a WF classifier and count the number of activation for each neuron, namely $C \in [0, p]$. We consider both most and least frequently activated neurons represent the features of the website. They can potentially stimulate misclassifications in DL-based WF classifiers. Hence, we propose the following two selection strategies and compare their effectiveness in experiments:

- Strategy 0: Select neurons that have been most frequently activated in the past.
- Strategy 1: Select neurons that have been least frequently activated in the past.

Since we need to increase the confidence of the mutated variant traffic, the second part of the expression $c_1(x) - c_0(x)$ is designed to maximize the gap between the probability of being classified as the correct class label and the highest probability of being misclassified. $c_0(x)$ signifies the probability of the seed x being classified as the correct class label, while $c_1(x)$ denotes the highest probability value among all incorrect class labels predicted by WF classifiers. If $c_1(x)$ is greater than $c_0(x)$, it indicates that a successful variant trace is found, which can mislead WF classifiers. Therefore, we try to maximize the second part in the joint optimization function to further increase the confidence. The variables λ_1 and λ_2 are used to balance these two parts. The values for these variables are determined through empirical experiments discussed in Section IV.

F. Trace Mutation

Trace mutation involves obtaining the gradients by computing the partial derivative of the objective function in terms of the input variable x (i.e., seed), and determining the injection positions and directions of dummy cells in each representative trace of a website based on the gradients. The gradient is in the form of a gradient vector, of which the dimension aligns with that of the input seed, as defined in the following:

$$\nabla = \frac{\partial obj(x)}{\partial x} \tag{2}$$

We employ the gradient ascent technique to maximize the objective function for increasing the neurons values and maximizing the confidence of the misclassifications of the DL-based WF classifiers.

WFGUARD adopts two mutation strategies, defined as $f(\nabla)$: (1) only injecting a dummy Tor cell from the Tor client to the exit node (i.e., inserting +1 into the trace), and (2) injecting a dummy Tor cell in either direction (i.e., inserting +1 or -1 into the trace). When only inserting +1, the index with the maximum value in the gradient vector is selected as the injection position. The processing of the gradient when only injecting +1 is defined as follows:

$$f_{+1}(\nabla) = \{sign(\nabla_j); j \mid Max(\nabla_j)\} \tag{3}$$

The $sign(\nabla_j)$ pertains to extracting the sign of the j^{th} $(0 \le j \le d)$ element of the gradient vector, where d represents the dimension of the gradient, which is the same as the dimension of the input trace.

In the second mutation strategy, WFGUARD can inject either +1 or -1. In such case, the index with the maximum absolute value in the gradient vector is chosen as the injection position. We insert +1/-1 if the gradient is positive/negative in the injection position. The processing of gradient is defined as follows:

$$f_{+1=-1}(\nabla) = \{sign(\nabla_j); j \mid Max(abs(\nabla_j))\} \tag{4}$$

where $abs()$ represents the function to obtain the absolute value of the gradient. As a result, the function $f(\nabla)$ generates the injection patterns, encompassing both the injection positions and the injection directions.

G. Dummy Cell Injection

After obtaining the injection patterns, a mutated trace (denoted as $x + f(\ )$) is generated by injecting dummy cell following the injection patterns as shown in Equation (5). It is essential to ensure that the injected cells comply with the constraint of traffic trace. That is, dummy cells should not be injected into the part of the padded cells in the current trace. By carefully handling the injection process while adhering to the constraint mentioned above, WFGUARD ensures that the generated mutated trace sample x aligns with the features of traffic trace, thereby effectively increasing the neurons values and misclassifications of the DL-based WF classifiers.

$$x = \text{inject}(x; f(\ )) \tag{5}$$

H. Variant Trace Verification

The mutated trace sample x is fed into the WF classifiers for classification prediction. If x successfully misleads the WF classifiers, x is a favorable mutation sample. However, if it fails to deceive classifiers and the maximum number of mutations for the trace is not reached, we update the objective function with x and continue mutating the trace following the same procedure. Otherwise, if it reaches the maximum number of mutations, we discard the trace and select a new seed from the seed pool for mutation.

IV. EXPERIMENTAL EVALUATION

We evaluate the effectiveness and efficiency of our WF-GUARD with extensive experiments. We implement WF-GUARD using TensorFlow-GPU 1.15.0 and Keras 2.3.1 frameworks. All experiments are conducted on Ubuntu 18.04 system and 6 various NVIDIA GPU cards, including 2 Tesla K80 and 4 1080Ti cards.

A. Dataset

We validate the effectiveness of WFGUARD with a dataset collected by Sirinam et al. [31]. This dataset is commonly utilized to evaluate the efficiency of DL-based WF attacks and defenses. For the closed-world scenario, the dataset includes the most popular 95 websites from Alexa, each consisting of 1000 traces. It is used to train the DF [31] and Var-CNN [3] classifiers, where the ratio of training, validation, and test sets is 8:1:1. The open-world dataset is composed of an unmonitored dataset and the monitored dataset used in closed-world scenario. The unmonitored dataset includes 40,000 websites, each with one trace. We fix the length of input trace as 5000 in all experiments.

B. Metrics

The metrics used to evaluate WFGUARD include the bandwidth overhead (BWO) and detection rate (DR). We do not evaluate time overhead since WFGUARD solely performs dummy cell injection on traces, resulting in no time overhead.

- BWO represents the ratio of the number of dummy Tor cells injected into the trace to the number of actual total Tor cells of the original trace. It is used to measure the defense overhead in both closed-world scenario and open-world scenario. The larger BWO introduced by the defense, the less efficient the defense is.
- DR is utilized to evaluate the effectiveness of the WF classifiers in correctly classifying traffic traces. A lower DR indicates higher defense effectiveness. DR is defined as the ratio of the number of traces correctly classified by attackers to the total number of traces.

C. Experimental Setup

The seed pool initialization, as described in Section III, involves experimentally selecting q representative traces that are accurately recognized by DL-based WF classifiers with a detection rate exceeding 95% for each website. Therefore, for all seeds in the seed pool, applying WFGUARD can result in at most q effective injection patterns for each website. The optimal value of q are determined through experiments.

As for the joint optimization function, we propose two neuron selection strategies: select neurons that have been activated the most in the past (Strategy 0) and select neurons that have been least frequently activated in the past (Strategy 1). To obtain the activation count of each neuron, WFGUARD feeds 100 traces of each website into DL-based WF classifiers and count the number of activation for each neuron, thus we have the activation count $C \in [0; 100]$.

The parameters, i.e., $_1$ and $_2$, in the joint optimization are weight coefficients that measure the importance of each objective, as shown in Equation (1). In order to balance the importance of neurons values and misclassification possibility in the joint optimization function, we set $_1 = \frac{1}{m}$ and $_2 =$ 1, where m represents the number of selected neurons, and is the threshold for activating neurons. The optimal value for is determined through empirical experiments.

WFGUARD aims to find an effective injection pattern for each of the 95 websites. Thus, for each website, we conduct experiments to further evaluate the effectiveness of some combinations of these q injection patterns, each of which derives from a single trace of the website. We first apply q injection patterns to the training dataset (800 traces). For each injection pattern, we feed the corresponding 800 injected traces into the DL-based WF classifiers to obtain detection rates. Since we have q injection patterns, q detection rates are obtained in total. In the first case denoted as WFGUARD-light, we combine the two injection patterns with top-2 smallest detection rates. In the second case denoted as WFGUARD-heavy, we combine the three patterns with top-3 smallest detection rates, which generates more bandwidth overhead and lower detection rate. We do not include the experiment results of only applying one injection pattern since it apparently results in high classification accuracy.

D. Experimental results

Baseline: WFGUARD first evaluates performance of the two WF classifiers, DF [31] and Var-CNN [3], in both the closed-world and open-world scenarios on the undefended dataset. The evaluation serves as the baseline for comparing with the

TABLE I
CLASSIFIERS RESULTS ON THE NO-DEFENDED DATASET IN THE CLOSED-WORLD (CW) AND OPEN-WORLD (OW) SETTINGS

Models	DF	Var-CNN
CW	98.35%	98.40%
OW	96.80%	97.23%

proposed defense in this paper. The results are presented in Table I. As we can see from Table I, in the closed-world scenario, DF and Var-CNN achieve high detection rate of 98.35% and 98.40% respectively. However, compared with that in the closed-world scenario, the DR decreases in the open-world scenario due to the significantly increased data size. Though the DR of two classifiers are still above 96%.

Parameters Tuning: A large number of experiments are conducted to tune the parameters used in WFGUARD, including , , and q. is a threshold used to measure whether a neuron is activated. is a coefficient, which is utilized to control the bandwidth overhead, q is the number of seeds for each website, which also determines the number of injection patterns that we can obtain from different mutation strategies.

The relationship between different and detection rate of DL-based WF classifiers is shown in Figure 4, in which the neuron selection strategy is Strategy 0 and the mutation strategy is "+1". Note that these strategies are also adopted in experiments corresponding to Figure 5 and Figure 6. Figure 4 illustrates that setting to 0.2 within the range of $[0.1, 0.5]$ in the closed-world scenario results in the minimum DR for the DF model, while setting to 0.3 in the open-world scenario also leads to the minimum DR. According to the results, is set to 0.2 in the closed-world scenario and 0.3 in the open-world scenario in the subsequent experiments. Figure 5 shows that within the range of $\in [5\%, 25\%]$, setting to 20% results in the minimum DR for the DF model. Figure 6 illustrates that within the range of $q \in [5, 25]$, setting q to 20 results in the minimum DR for the DF model results in the optimal performance.

Closed-world experiment results analysis: After determining the optimal threshold $= 0.2$, we first explore the impact of selecting different numbers of neurons on the detection rate of the DF and Var-CNN models under different neuron selection strategies. As shown in Figure 7, when using neuron selection Strategy 0, WFGUARD shows better defense performance on the DF and Var-CNN models compared to Strategy 1. Therefore, we determine Strategy 0 as our neuron selection strategy.

Then, further exploration of the defense effectiveness with different mutation strategies is conducted. Figure 8 shows the impact of two different mutation strategies on the detection rate of DF and Var-CNN models. In both situations, only injecting +1 outperforms the other strategy.

Figure 9 illustrates the generalization of WFGUARD, which is reflected by its ability to reduce the detection rate of an unknown model when injecting the injection patterns derived from attacking the known model with known structure. Then the patterns are injected the patterns into the original traffic and fed into the unknown classifier. To validate the generalization of WFGUARD, the injection patterns obtained from experiments on DF model, where the neuron selection strategy is Strategy 0 and the number of neurons is 50 according to Figure 7, are used to defend the Var-CNN model. As shown in Figure 9, the results demonstrate that WFGUARD effectively reduces the detection rate of the Var-CNN model, regardless of whether the mutation strategy is only +1 or +1/-1. Particularly, in the case of +1/-1 injection, the injection patterns generated from the DF model prove to be more effective to the DF model than to the Var-CNN model.

Table II presents a comparison of defense effectiveness and overhead among different defense methods in the closed-world scenario. The results in the table indicate that WFGUARD-light achieves a bandwidth overhead of 14.18% to reduce the detection rate of DF and Var-CNN to below 8.8%. This is approximately 14% lower in bandwidth overhead compared to the BAND defense method, with a similar defense effectiveness to BAND. On the other hand, WFGUARD-heavy outperforms BAND with over 4% lower bandwidth overhead while achieving better defense effectiveness. Furthermore, when compared to Mockingbird and WTF-PAD, both WFGUARD-light and WFGUARD-heavy achieve significantly lower DRs with bandwidth overhead reduced by 10% to 40%. The reduction in DRs ranges from around 30% to 82%, demonstrating the superiority of WFGUARD over existing defense methods.

TABLE II
COMPARISON OF WFGUARD WITH OTHER DEFENSE METHODS IN THE CLOSED-WORLD SCENARIO.

Models \ Methods		WFGUARD-light	WFGUARD-heavy	BAND	Mocking Bird	WTF-PAD
DF	BWO	14.18%	21.43%	25.02%	58.02%	63.23%
	DR	8.80%	5.62%	5.12%	38.11%	90.85%
Var-CNN	BWO	11.04%	15.32%	25.07%	58.12%	63.12%
	DR	4.43%	2.15%	1.51%	35.21%	94.02%

Open-world experiment results analysis: We examine the impact of different neuron selection strategies on the detection rate of the DF and Var-CNN models in the open-world scenario. As shown in Figure 10, when neuron selection Strategy 0 is adopted, the defensive effect of WFGUARD on both the DF and Var-CNN models is superior to when Strategy 1 is used. The WFGUARD, when using Strategy 0, effectively reduces the detection rate of both DL-based WF classifiers to their minimum values. By setting the number of neurons to 30 and 50 respectively, WFGUARD reduces the detection rate of the DF and Var-CNN attack models to their lowest values of 10.73% and 6.00% respectively.

The aforementioned results assume that the mutation operation is only injecting +1. To fully validate the WFGUARD defense method, it is necessary to further explore the defensive effects of different mutation strategies. Figure 11 shows the impact of two different mutation strategies on the detection rate of the DF and Var-CNN models under neuron selection Strategy 0 in the open-world scenario. The results indicate

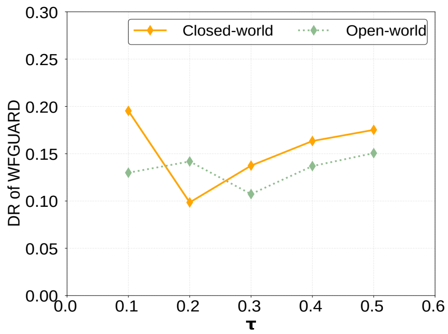

Fig. 4. Different selection of τ

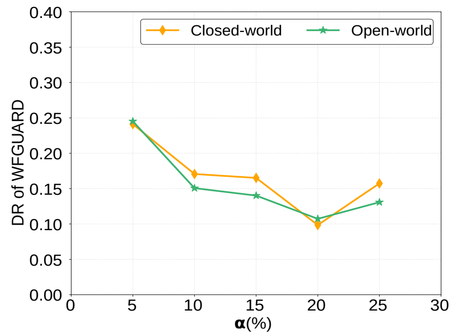

Fig. 5. Different selection of α

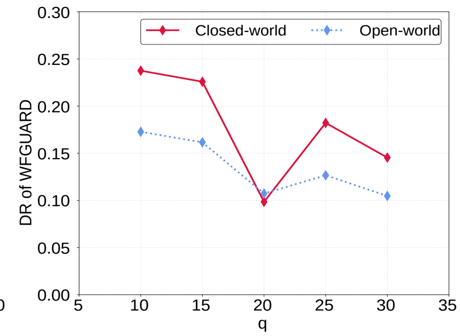

Fig. 6. Different selection of q

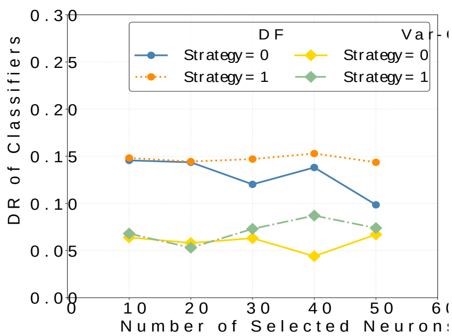

Fig. 7. Different neuron selection strategies in the closed-world scenario.

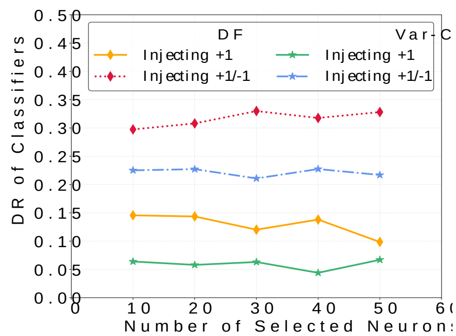

Fig. 8. Different mutation operations in the closed-world scenario.

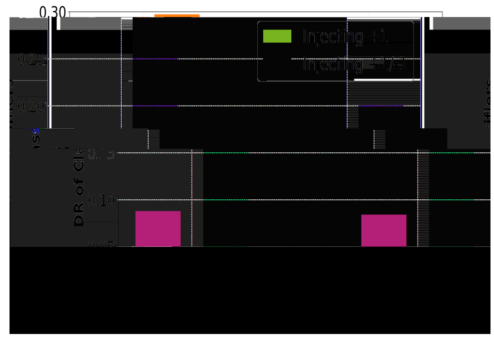

Fig. 9. The generalization of WFGUARD in the closed-world scenario.

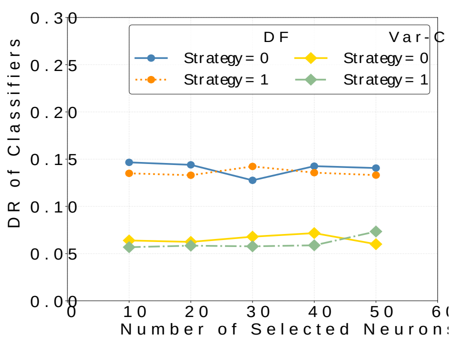

Fig. 10. Different neuron selection strategies in the open-world scenario.

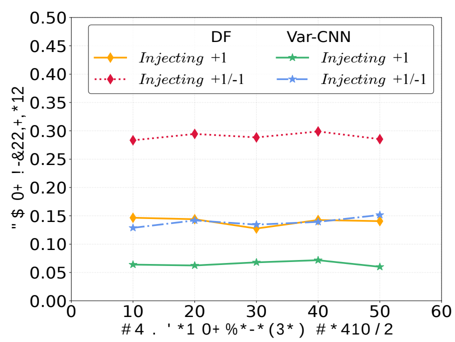

Fig. 11. Different mutation operations in the open-world scenario.

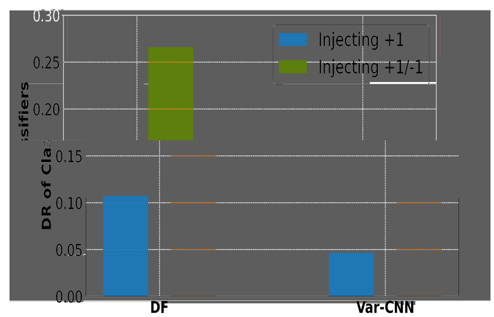

Fig. 12. The generalization of WFGUARD in the open-world scenario.

that WFGUARD shows better performance on both the DF and Var-CNN models when only injecting +1, as compared to injecting +1/-1 . Furthermore, the detection rate of the DF model can be reduced to the lowest level when the number of neurons is 50 and only +1 is injected.

Similar to the closed-world scenario, we also validated the generalization of WFGUARD in the open-world scenario, and the results are depicted in Figure 12. No matter only injecting +1 or injecting +1/-1 is used as the mutation strategy, WFGUARD can effectively reduce the detection rate of the Var-CNN model, especially when injecting +1/-1 is adopted, and the injection pattern generated from the DF model effectively reduces the detection rate of the Var-CNN model to 10.7%. The results indicate that in both open-world and closed-world scenarios, the WFGUARD defense method is generalized and deceive unknown models.

Table III summarizes the comparison between WFGUARD and other existing defense methods in terms of BWO and DR in the open-world scenario. The results indicate that WFGUARD-light can reduce the detection rate of DF and Var-CNN to below 10.73% with a BWO of 14.18%, which is 40% and 30% lower in BWO and DR respectively than Surakav-light. Compared with Surakav-heavy, WFGUARD-heavy can achieve better defense effects with about 55% less bandwidth overhead. Compared with WTF-PAD, the WFGUARD defense method even achieves a lower detection rate of about 81% with 13% less bandwidth overhead. All the results show that WFGUARD is superior to existing defense methods.

V. RELATED WORK

This section provides an overview and brief analysis of current WF attacks and WF defenses.

TABLE III

C

References

[1] K. Abe and S. Goto. Fingerprinting attack on Tor anonymity using deep learning. *Proceedings of the Asia Pacific Advanced Network (APAN)*, 42:15–20, 2016.

[2] A. Abusnaina, R. Jang, A. Khormali, D. Nyang, and D. Mohaisen. DFD: Adversarial Learning-based Approach to Defend Against Website Fingerprinting . In *Proceedings of the 39th IEEE International Conference on Computer Communications (INFOCOM)*, pages 2459–2468, 2020.

[3] S. Bhat, D. Lu, A. Kwon, and S. Devadas. Var-CNN: A Data-Efficient Website Fingerprinting Attack Based on Deep Learning. In *Proceedings on Privacy Enhancing Technologies (PET)*, volume 2019, pages 292–310, 2019.

[4] X. Cai, R. Nithyanand, and R. Johnson. CS-BuFLO: A Congestion Sensitive Website Fingerprinting Defense. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES)*, pages 121–130, 2014.

[5] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg. A systematic approach to developing and evaluating website fingerprinting defenses. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 227–238, 2014.

[6] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson. Touching from a Distance: Website Fingerprinting Attacks and Defenses. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 605–616, 2012.

[7] W. De la Cadena, A. Mitseva, J. Hiller, J. Pennekamp, S. Reuter, J. Filter, T. Engel, K. Wehrle, and A. Panchenko. Trafficsliver: Fighting website fingerprinting attacks with traffic splitting. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1971–1985, 2020.

[8] 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 *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, pages 332–346, 2012.

[9] J. Gong and T. Wang. Zero-delay Lightweight Defenses against Website Fingerprinting. In *Proceedings of the USENIX Security Symposium (Security)*, pages 717–734, 2020.

[10] J. Gong, W. Zhang, C. Zhang, and T. Wang. Surakav: generating realistic traces for a strong website fingerprinting defense. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1558–1573. IEEE, 2022.

[11] J. Guo, Y. Jiang, Y. Zhao, Q. Chen, and J. Sun. Dlfuzz: Differential fuzzing testing of deep learning systems. In *proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE/ESEC)*, pages 739–743, 2018.

[12] J. Hayes and G. Danezis. k-fingerprinting: a Robust Scalable Website Fingerprinting Technique. In *Proceedings of the USENIX Security Symposium (Security)*, pages 1187–1203, 2016.

[13] S. Henri, G. García, P. Serrano, A. Banchs, P. Thiran, et al. Protecting against website fingerprinting with multihoming. *Proceedings on Privacy Enhancing Technologies*, 2020(2):89–110, 2020.

[14] 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 Symposium on Computers and Communications (ISCC)*, pages 1–7. IEEE, 2020.

[15] N. Humbatova, G. Jahangirova, and P. Tonella. Deepcrime: Mutation testing of deep learning systems based on real faults. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*, pages 67–78, 2021.

[16] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright. Toward an Efficient Website Fingerprinting Defense. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*, volume 9878, pages 27–46, 2016.

[17] D. Li, Y. Zhu, M. Chen, and J. Wang. Minipatch: Undermining dnn-based website fingerprinting with adversarial patches. *IEEE Transactions on Information Forensics and Security (TIFS)*, 17:2437–2451, 2022.

[18] Z. Ling, J. Luo, K. Wu, W. Yu, and X. Fu. TorWard: Discovery of Malicious Traffic over Tor. In *Proceedings of the 33rd IEEE International Conference on Computer Communications (INFOCOM)*, 2014.

[19] Z. Ling, J. Luo, D. Xu, M. Yang, and X. Fu. Novel and Practical SDN-based Traceback Technique for Malicious Traffic over Anonymous Networks. In *Proceedings of the 38th IEEE International Conference on Computer Communications (INFOCOM)*, pages 1180–1188, 2019.

[20] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, and W. Jia. A New Cell Counting Based Attack Against Tor. In *Proceedings of the ACMConference on Computer and Communications Security (CCS)*, pages 578–589, November 2009.

[21] X. Luo, P. Zhou, E. W. Chan, W. Lee, R. K. Chang, R. Perdisci, et al. Httpos: Sealing information leaks with browser-side obfuscation of encrypted flows. In *NDSS*, volume 11, 2011.

[22] M. Nasr, A. Bahramali, and A. Houmansadr. Defeating *f*DNN-Based*g* traffic analysis systems in *f*Real-Time*g* with blind adversarial perturbations. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2705–2722, 2021.

[23] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel. Website Fingerprinting at Internet Scale. In *Proceedings of the Network Distributed System Security Symposium (NDSS)*, 2016.

[24] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website Fingerprinting in Onion Routing based Anonymization Networks. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 103–114, 2011.

[25] K. Pei, Y. Cao, J. Yang, and S. Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18, 2017.

[26] M. S. Rahman, M. Imani, N. Mathews, and M. Wright. Mockingbird: Defending Against Deep-LearningBased Website Fingerprinting Attacks With Adversarial Traces. *IEEE Transactions on Information Forensics and Security (TIFS)*, pages 1594–1609, 2020.

[27] M. S. Rahman, P. Sirinam, N. Mathews, K. G. Gangadhara, and M. Wright. Tik-Tok: The Utility of Packet Timing in Website Fingerprinting Attacks. In *Proceedings on Privacy Enhancing Technologies (PET)*, pages 5–24, 2020.

[28] V. Rimmer, D. Preuveneers, M. Juarez, T. V. Goethem, and W. Joosen. Automated Website Fingerprinting through Deep Learning. In *Proceedings of the Network Distributed System Security Symposium (NDSS)*, 2018.

[29] A. M. Sadeghzadeh, B. Tajali, and R. Jalili. Awa: Adversarial website adaptation. *IEEE Transactions on Information Forensics and Security (TIFS)*, 16:3019–3122, 2021.

[30] M. Shen, K. Ji, Z. Gao, Q. Li, L. Zhu, and K. Xu. Subverting website fingerprinting defenses with robust traffic representation.

[31] P. Sirinam, M. Juarez, M. Imani, and M. Wright. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 1928–1943, 2018.

[32] The Tor Project, Inc. Tor: Anonymity Online. https://www.torproject.org/, 2023.

[33] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg. Effective Attacks and Provable Defenses for Website Fingerprinting. In *Proceedings of the USENIX Security Symposium (Security)*, pages 143–157, 2014.

[34] T. Wang and I. Goldberg. Improved Website Fingerprinting on Tor. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 201–212, 2013.

[35] T. Wang and I. Goldberg. Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In *Proceedings of the USENIX Security Symposium (Security)*, pages 1375–1390, 2017.

[36] X. Xie, L. Ma, F. Juefei-Xu, M. Xue, and et.al. Deephunter: A coverage-guided fuzz testing framework for deep neural networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*, pages 146–157, 2019.

[37] P. Zhang and a. Q. D. Bin Renand Hai Dong. Cagfuzz: Coverage-guided adversarial generative fuzzing testing for image-based deep learning systems. *IEEE Transactions on Software Engineering (TSE)*, 48(11):4630–4646, 2021.

[38] Z. Zhuo, Y. Zhang, Z.-l. Zhang, X. Zhang, and J. Zhang. Website fingerprinting attack on anonymity networks based on profile hidden markov model. *IEEE Transactions on Information Forensics and Security (TIFS)*, 13(5):1081–1095, 2017.