SAD: Website Fingerprinting Defense Based on Adversarial Examples

Renzhi Tang[1,2], Guowei Shen[1,2(✉)], Chun Guo[1,2], and Yunhe Cui[1,2]

[1] College of Computer Science and Technology, Guizhou University, Guiyang 550025, China
gwshen@gzu.edu.cn

[2] Guizhou Provincial Key Laboratory of Public Big Data, Guiyang 550025, China

Abstract. Website Fingerprinting (WF) attacks can infer website names from encrypted network traffic when the victim is browsing the website. Inherent defenses of anonymous communication systems such as The Onion Router (Tor) cannot compete with current WF attacks. The state-of-the-art attack based on deep learning can gain over 98% accuracy in Tor. However, the existing defense will bring high bandwidth overhead, affect the user's network experience or cannot be used in actual scenarios. Some researchers found that deep learning models are vulnerable to adversarial examples. In this paper, based on adversarial examples we propose Segmented Adversary Defense (SAD) for deep learning-based WF attacks. Network traffic is divided into multiple segments. Then, the adversarial examples for each segment of traffic can be generated by SAD. Finally, dummy packets from adversarial examples are inserted after each segment traffic. Experimentally, our results show that SAD can effectively reduce the accuracy of WF attacks. The technique drops the accuracy of the state-of-the-art attack hardened from 96% to 3% while incurring only 40% bandwidth overhead. Compared with the existing proposed defense named Deep Fingerprinting Defender (DFD), the defense effect of SAD is better under the same bandwidth overhead.

Keywords: Website fingerprinting attack · Website fingerprinting defense · Tor · Adversarial examples

1 Introduction

Anonymous communication system can protect the users' information through data encryption and multi-hop proxy. The Onion Router (Tor) [6], a kind of anonymous communication software, is widely used worldwide. However, inherent defense of Tor is difficult to against WF attacks. In WF attack, encrypted traffic is collected by WF attacker between client and guard node of Tor. Then, the website name can be infer from the traffic obtained by attackers.

W. Shi et al. (Eds.): SPNCE 2021, LNICST 423, pp. 88–102, 2022.
https://doi.org/10.1007/978-3-030-96791-8_7

In the past, the attacker extracts a variety of packets level features from encrypted traffic, such as number, order, direction and unique length, then use machine learning for classification [8,16,22]. In recent years, deep learning has been recognized in various fields such as image classification [15] and speech recognition [11]. Different from traditional machine learning, automatic extraction of features from original data is one of the advantages of deep learning. WF attacks based on deep learning have been proposed such as Automated Website Fingerprinting (AWF) [19], Var-CNN [2] and Deep Fingerprinting (DF) [20]. DF can gain more 98% accuracy only using direction sequence of traffic.

With the attacks are constantly updated, some WF defenses have been proposed such as WTF-PAD [14], Walkie-Talkie (WT) [23] and Deep Fingerprinting Defender(DFD) [1]. In addition, researchers have found that deep learning models are vulnerable to adversarial examples [7]. It can make a classifier based on deep learning misclassify [5]. Therefore, some WF defenses based on adversarial examples is proposed such as WF-GAN [12] and Mockingbird [18]. But, WTF-PAD, WT and DFD have expired defense capability and suboptimal bandwidth overhead. WF-GAN and Mockingbird are difficult to apply to live traffic. The reason is that they need adversarial examples based on an integral traffic flow. The situation is unrealistic.

In response to the above problems, we propose a defense SAD based on adversarial examples. SAD consists of two components: an adversarial examples generation model and a WF attack model. The former can generate adversarial examples according to direction sequence of traffic flow. The WF attack model provides an optimization scheme for SAD during training. Experiments show that SAD effectively reduces the attack success accuracy of DF, Var-CNN and AWF attack models with slight bandwidth overhead. SAD have excellent defense capabilities whether facing unknown attack or new adversarial training. The main contributions of this paper are as follow:

1) This paper proposes a WF defense, named SAD, using segmented processing and adversarial examples. Compared with some previous studies, SAD does not require global traffic sequence information when generating adversarial examples. Therefor, It can apply in live traffic.
2) SAD can be easily generated adversarial examples for different attack models based on deep learning. The input and output of SAD are the original direction sequence data rather than burst-based data. This is same data format required by the attack model.
3) SAD can effectively reduce the attack success rate, even facing adversarial training or unknown attack. For the three existing WF attack models, DF, Var-CNN and AWF, SAD can reduce these attack success rate from 96% to 1%. Compare with DFD, we have better defense whether in open-world or close-world.

2 Related Work

2.1 Website Fingerprinting Attack

Herrmann et al. [10] first applied the WF attack on Tor in 2009, but the accuracy was only 3% in close-world. In 2011, Panchenko et al. [17] proposed a WF attack

based on Support Vector Machines (SVM). It improves the accuracy of WF attacks. But, the computational cost of these WF attack classifiers is too high and not practical. With the development of computers, researchers have proposed more effective attacks. In 2014, the K-Nearest Neighbors classifier proposed by Wang et al. [22]. The attack achieved higher accuracy with shorter training time and test time. In addition, Panchenko et al. [16] used the cumulative sum of packet lengths as features and SVM as a classifier, which achieved good results in close-world.

In order to overcome the shortcomings of WF attack methods based on traditional machine learning, researchers have leveraged deep learning. To skip the process that manually extracting features, Rimmer et al. [19] proposed a deep learning-based WF attack AWF. They collected a lot of training data and tried to use three different neural network structure, Stacked Denoising Autoencoder (SDAE), Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM). The action make accuracy achieve 96%. Sanjit [2] proposed Var-CNN based on ResNets [9] deep learning. It has a better recognition effect when the amount of training data is small, especially in close-world. Sirinam et al. [20] proposed a CNN-based WF attack model DF. The model has reached a high level of attack ability in both close-world and open-world.

2.2 Website Fingerprinting Defense

WF defense against WF attacks by inserting dummy packets. The defense will bring extra bandwidth overhead that affects the user's experience. All defenses aim at high misclassification rate and low bandwidth overhead. BuFLO [6] sends packets at the same constant rate in both directions of the network traffic, and ends the defense of packets within a short period of time after the page is loaded. But, it brings unreasonable bandwidth overhead. Tamaraw [4] and CS-BuFLO [3] extend BuFLO. The difference is that the download and upload rates can be set to different rates. But their bandwidth overhead exceeds 100%. The WTF-PAD proposed by Juarez et al. [22]. The defense attempts to fill a large time gap between two data packets. Whenever the gap is large, WTF-PAD will send a series of dummy packets. It reduce the accuracy of K-NN attacks. However, WTF-PAD was defeated by the WF attack DF. Walkie-Talkie (WT) [23], makes the attack success rate of DF only 49.7% with 31% bandwidth overhead. WT changed the communication mode of both parties from full-duplex to half-duplex. The change affects the user's network experience. For WF attack based on deep learning, some researchers proposed WF defense based on adversarial examples, WF-GAN [12] and Mockingbird [18]. They have excellent performance, but they difficult to be applied in live traffic due to the requirements of model input data.

The latest defense method DFD [1] injects a dummy burst sequence according to the previous burst sequence in the real-time traffic. When the injection method is the server-side injection method, the attack success rate using the convolutional neural network (CNN) as the attack model is reduced from 99.93% to 5% with 85.56% bandwidth overhead. However, he did not test the existing WF attacks. In addition, the DFD cannot stably insert dummy packets. In

some cases, the number of inserted dummy packets is small. When bandwidth overhead is small, the expected defense effect cannot be achieved.

3 Preliminary

3.1 Threat Model

Tor protects browsing website information of user through data encryption and multi-hop proxy. Its working principle is shown in Fig. 1. In general, User randomly select three relay nodes from global active relay nodes. These selected nodes will form a communication link. All traffic will be encrypted when through the link. The purpose of WF attacker is to obtain the website name visited by users from encrypted traffic. To be specific, the attacker monitors and saves traffic between the guard node and user. Then, the attacker uses the traffic to infer the website name using deep learning. The attacker generally need direction sequence of traffic packet. The outgoing (client to server) and incoming (server to client) packets are represented as $+1$ and -1, respectively.

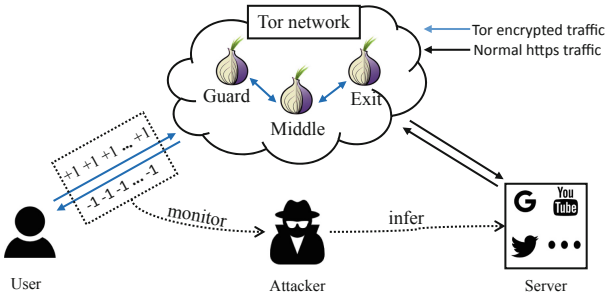

Fig. 1. WF attack mode.

However, the attacker did not own all the data of the website. We mark target websites as monitored sites, and other sites as unmonitored sites. To this end, WF attack and WF defense are evaluated in two different experimental environments: close-world and open-world. Close-world means that users can only access monitored websites. The open-world is closer to the real scene than the close-world. It allow user to access any website.

3.2 Defense Model

The goal of the defender is to reduce the accuracy of WF attacks. One of the best ways to reduce the accuracy is to change the features the traffic. For this, there are two operation: inserting dummy packets and delay packets. Considering the user's network experience, inserting dummy packets usually is adopted by WF

defense. The user or the guard node only decide to send packets in a single direction, dummy packets senders are deployed at both ends. In addition, there are two scenarios for WF defense: black-box or white-box. Black-box means that the structure and parameters of the WF attack model are unknown. We only know the correspondence between the input and output of the model; white-box means that the structure and parameter information of the attack model can be obtained. We can interact deeply with the attack model.

3.3 Adversarial Examples

Adversarial examples is a carefully designed input data for deep learning model, which will cause the model to output incorrect prediction results. It was first discovered by the research in the field of image classification [21]. Fast Gradient Sign Method (FGSM) [7] is a classic adversarial examples generation method based on gradient. It can be expressed as $x' = x + \alpha Sign(\nabla_x J(\theta, x, y))$, x' is adversarial examples, θ is the parameter of deep learning model, x is the input of the model, y is the correct category for x, $J(\theta, x, y)$ is loss function, $\nabla_x J(\theta, x, y)$ is the gradient of the loss function with respect to x, α is super parameter. The WF defense generates adversarial examples is similar to FGSM. WF-GAN [12] and Mockingbird [18] modifies burst-based features to generate adversarial examples. Due to the practical significance of the feature, the feature value can only increase but not decrease. Adversarial examples can expressed as $[b_1+\delta_1, b_2+\delta_2, ..., b_n+\delta_n]$, b_i is burst sequence of original flow, δ_i is perturbation size and nonnegative number that means can't drop packet.

4 Defense Design

In this paper, adversarial examples will be generated by SAD. The role of adversarial examples is to mislead the attack model. The purpose for protecting user's privacy information is achieved by using adversarial examples. Misleading attack models are an intuitive way of defending. Figure 2 represents the overall flow of SAD. x_i is direction sequence that is input data of SAD. y_i is the dummy packets being inserted. Adversarial examples $x' = [x_1, y_1, x_2, y_2, ..., ..., x_i, y_i]$ is output of SAD. We assume C is attack model, L is predict label of the attack model. In that way, SAD makes $C(x') \neq C(x) = L_x$, where $x' = x \oplus y$, y is dummy packets in x', $\oplus$ indicates how to insert the dummy packets. The number of packets inserted is proportional to the increased bandwidth overhead. So, we want to minimize the number of dummy packets when the defense effect remains unchanged.

4.1 SAD Model Design

This paper uses the solution for injecting dummy packets at both ends. At the same time, SAD has the characteristics of low bandwidth overhead and diversified adversarial examples. The generating of adversarial examples is shown in Fig. 3.

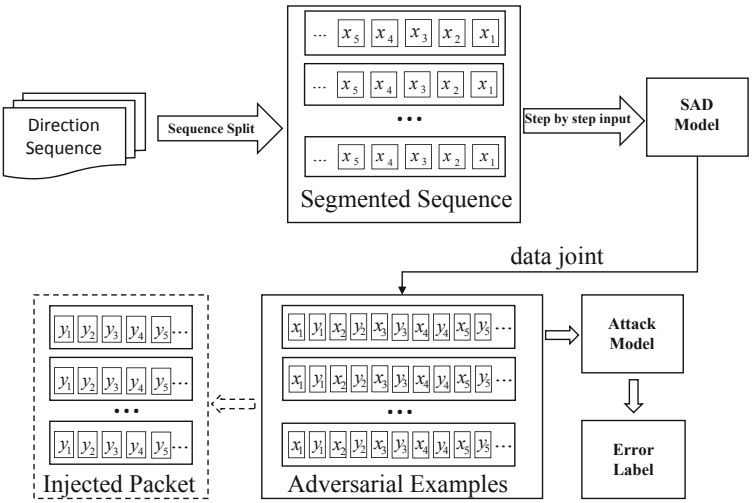

Fig. 2. SAD defense process

Packets Direction Sequence Segmentation. The segmentation refers to cutting the sequence into different lengths according to different segmented packet number (SPN). SPN set ($SPNS$) consists of several different SPN. To public data set, each piece of data is a packets directions sequence generated during the complete access process of a certain website. The purpose of segmentation is to simulate the transmission process of packets. The segmented adversarial examples y_i is inserted after each segment of original packets until the remaining length of sequence is less than $min(SPNS)$. Finally, the segmented sequence and the segmented adversarial examples are sequentially connected to form a complete adversarial examples.

Generating Adversarial Examples. SAD contains multiple sub-models with similar structures. The number of sub-models is equal to the number of preset $SPNS$ elements. The sub-model structure is a simple multi-layer fully connected neural network. Its activation function is $Relu$, as in Formula 1. In order to shorten the model training time and improve the robustness of the model, $Dropout$ and $Batch\ Normalization$ are used. The input of the sub-model is direction sequence, specifically $+1$ or -1. The features dimension of the sub-model output is $\lfloor SPN * BO \rfloor$ for strictly control the bandwidth overhead, where BO is bandwidth overhead. Significantly, output of sub-model only contain -1 or $+1$, We applied $Tanh$ and $Sign$ to the model output. $Tanh$ and $Sign$ as in Formula 2 and Formula 3. Finally, output of sub-model represent the direction of dummy packets. The main function of the sub-model is to generate adversarial examples corresponding to the segmented sequence. The output of multiple sub-models is spliced with segmented sequence to generate complete adversarial examples, the process is shown in Algorithm 1.

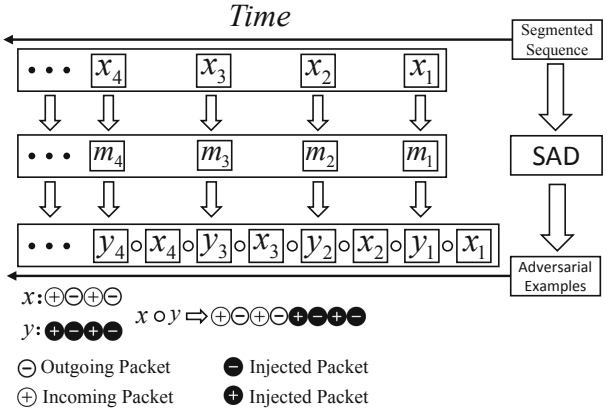

Fig. 3. SAD generates adversarial examples

$$Relu(x) = max(0, x) \tag{1}$$

$$Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2}$$

$$\text{Sign}(x) = \begin{cases} -1, x < 0 \\ 1, x \geq 0 \end{cases} \tag{3}$$

In Algorithm 1, The input dimensions of sub-model in the SMS correspond to the $SPNS$ elements one-to-one. First, we randomly select SPN from $SPNS$. Second, we choose the corresponding sub-model m_i that input dimension is SPN. Finally, the sub-sequence $x_i(length = SPN)$ is selected from f and input into the selected sub-model to obtain the output y_i. In other words, the first segmented sequence x_1 is put into the sub-model m_1 to get output y_1. The input of the sub-model m_2 selected for the second time is x_2, and the corresponding output is y_2. By analogy, the input of the sub-model m_i is x_i, and the output is y_i. Finally, the adversarial examples generated by all sub-models are spliced with the segmented sequence to generate adversarial examples y, in the form of $[x_1, y_1, x_2, y_2, ..., x_{i-1}, y_{i-1}, x_i, y_i]$.

SAD Model Training. First, SAD outputs the adversarial examples according to the original data. Then, *loss* will be generated when the adversarial examples are fed to the attack model, loss function is $CrossEntropyLoss$. The difference is that our goal is to make the loss larger. Moreover, we only update the parameters of SAD. In this way, the accuracy of the attack model will decrease with adversarial sample generated by SAD. The attack model can be any WF attack model based on deep learning. Significantly, we use $Sign$ to map the SAD output

Algorithm 1: SAD generate adversarial examples

Input: Packets direction sequence: f; Sub-model set: SMS; Segment packet number set: $SPNS$
Output: Adversarial examples: AE

Define: ODL : sequence f length; SI : split index of f; RL : remaining length of f; Res : a list; CSS : current segment sequence; CSM : current sub-model; SAE : segment adversarial examples; LSS: last segment sequence

```
while RL ≥ min(SPNS) do // RL initial value is ODL
    Initialize CandidateSPN to an empty set
    for spn in SPNS do
        if spn ≤ RL then
            CandiiateSPN.addElement(spn)
        end
    end
    SPN ← random choice a element from CandidateSPN
    CS ← split f index from SI to SI + SPN
    SI ← SI + SPN, RL ← RL - SPN
    CSM ← choice a sub-model from SMS according to SPN
    SAE ← CSM(CSS) // CSM compute SAEusing CSS
    Res.addElement(CSS, SAE)// Res initial value is empty array
end
if RL > 0 then
    LSS ← split f index from ODL-RL to end
    Res.addElement(LSS)
end
AE ← joint Res elements in order
```

to -1 or $+1$. But this function is not diversified, the input gradient of $Sign$ as the output gradient.

5 Experiment and Analysis

5.1 Dataset

In this paper, the experimental data set is collected by Sirinam et al. [20], which uses tor-browser-crawler [13] to simulate the process of users visiting the website. They visited Alexa 100 sites in the close-world, each website was visited 1250 times. In this process, tcpdump stores traffic as file in each browse. Next, they only store the data for at least 1000 valid visits after filtering out the invalid data. In the final result, 95 websites are stored. In the open-world, they visited 50000 Alexa websites, excluding the 100 websites in the close-world data set. Finally, they got 40716 unmonitored website traffic data after filtering invalid data.

These traffic can be expressed as a sequence of ($timestamp$, $packet\_size$, $direction$), where $direction$:$+1(-1)$ means outgoing (incoming) packet. Wang et

al. [22] ignored timestamps and packet size and only remain packet direction. On this foundation, Sirinam et al. [20] verification showed that using packet length does not get a attractive improvement in the accuracy of the attack. Therefore, all data is represented by a sequence of packet directions. The sequence contains only -1 and $+1$.

In order to evaluate the defense performance of DFD and compare it with SAD, the data is used to generate defense data using server-side injection according to the algorithm in DFD. The average bandwidth overhead range of these defense data is $0 < BO \leq 1$.

5.2 Performance Index

In this paper, we define three performance index: Attack Success Rate (ASR), Defense Success Rate (DSR) and Bandwidth Overhead (BO), as shown in Formula (4)–(6):

$$ASR = \frac{N_1}{N} \tag{4}$$

$$DSR = \frac{N_2}{N} \tag{5}$$

$$BO = \frac{I_{total}}{OV_{total}} \tag{6}$$

In Formula 4 to 6, N is the number of browsing records contained in the data set, N_1 is the number of data correctly identified by the WF attack model, N_2 is the number of error identification of the WF attack model, I_{total} is the number of inserted dummy packets, OV_{total} is the number of valid packets of the source data.

5.3 Result and Discussion

(1) WF Attack Model Evaluation in Different Sequence Length. Deep learning-based WF attacks use the direction sequence of packets as input data. The strategy adopted by WF defense is to insert dummy packets into normal traffic. The most obvious change brought by this strategy is that the length of the direction sequence becomes longer. In previous WF attacks based on deep learning, the sequence length was usually set to 5000. If the sequence length is less than 5000, fill it with 0, if it exceeds, it will be truncated directly. When the sequence length increases, the truncation length of the direction sequence is increased to obtain a credible result. Considering the range of bandwidth overhead $0 < BO \leq 1$, the sequence length is set $5000 \leq SecquenceLength \leq 10000$, the stride is 500. In this experiment, three WF attack models, DF, Var-CNN and AWF, are used in both close-world and open-world. The result is shown in Fig. 4.

It can be seen from Fig. 4 in close-world, different sequence length have stable ASR. When the sequence length is 7500, ASR of DF reaches the highest, 98.14%.

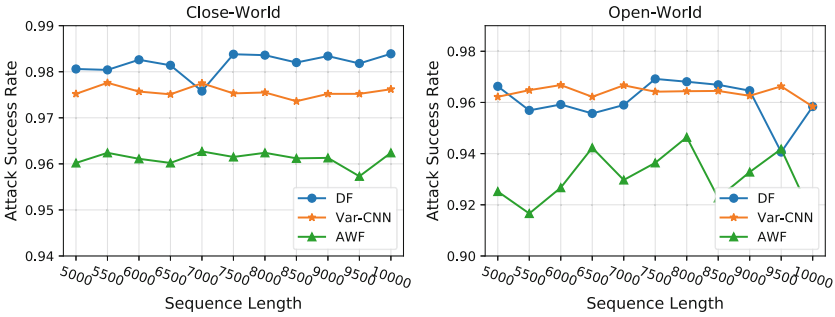

Fig. 4. Evaluation of website fingerprint attack model based on deep learning

When the sequence length is 7000, ASR of Var-CNN reaches the highest, 97.56%, which is in line with the ability performance in its original paper. In open-world, ASR of Var-CNN, AWF and DF reaches the highest, 96.39%, 91.28%, 96.04% respectively. In comparison, DF and Var-CNN are relatively stable. Therefore, in subsequent experiments related to AWF, ASR except AWF in open-world is between 96% and 97%. Finally, we can consider that the performance of the three attack models is stable in the current sequence length range.

(2) Defense Model Evaluation in Different BO. This experiment $spn \in \{10, 20, 30, 45, 40, 60, 100\}$. In different environments, the relationship between BO and DSR is shown in Fig. 5. It can be seen from Fig. 5 that DSR has been significantly improved with the increase of BO. In close-world, DSR for three attack models is above 90% when BO is 30%. DSR is close to saturation when BO is 40%. Although ASR of the three attack models in this experiment is 96%, there are differences in the defense capabilities of SAD for the three. When BO is 10%, DSR against Var-CNN is only 16.77%. With the same BO, DSR in open-world is higher. It can be seen that it is more beneficial to the attacker in close-world. It is more beneficial to the defender in open-world. In addition,

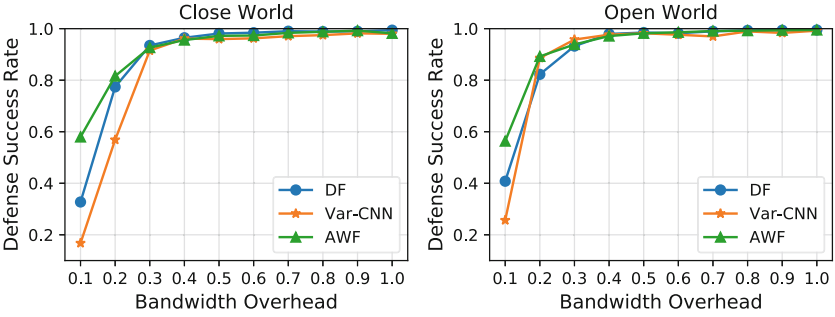

Fig. 5. SAD defense effect

DSR cannot be significantly increased when BO reaches a threshold. In practice, we can ignore unmonitored website and only confuse monitoring websites. This can save computing costs.

(3) Influence of SPNS. In the process of segmentation, the segment length is randomly selected from a certain SPNS, which increase the randomness of generating adversarial examples. To explore defense capabilities of SAD under different SPNS, we considered different SPNS on the DF attack model, as shown in Table 1. DSR under different SPNS is shown in Fig. 6.

Table 1. SPN set

$SPNS$ number	$SPNS$
1	{10, 20, 30, 45, 40, 60, 100}
2	{12, 16, 20, 35, 50, 75, 96}
3	{10, 20, 30, 45}
4	{40, 60, 100}
5	{100, 110, 130, 135, 150, 160}
6	{190, 200, 210, 215, 230, 250}

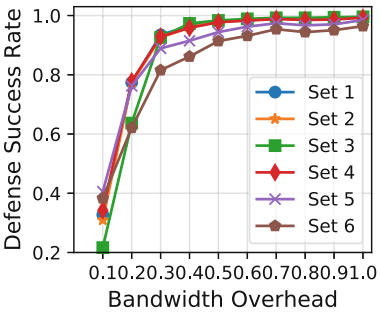

Fig. 6. Influence of different SPNS to DSR

In general, BO is proportional to DSR. DSR is basically same under different SPNS. The minimum value of each SPN set must consider the value of BO, because the output feature dimension of the sub-model is $\lfloor SPN * BO \rfloor$. Its value must be not less than 1 to create a compliant sub-model. The number of elements should not be too much. The number of sub-model and set elements is the same. If the model is too complex, it will increase difficulty for training SAD. On the contrary, if the number of set elements is too small, the model will not achieve a good defense effect because the model is not complex enough. We

deliberately provide set 5 and 6, which contain larger element values than other sets. This means that the length of segment sequence is longer, which leads to a certain degree of decline to DSR. We can obtain a conclusion that defense capability is related to segment length. From the experimental results, small segments are more favorable for defense.

(4) Compared with DFD. We use the algorithm that injecting dummy packets proposed by DFD to generate defense data. Then, we calculate the average of BO based on the defense data. DF is selected as the attack model. The purpose is that compared SDA and DFD on same original data. The results are shown in Fig. 7. In different experimental environments, DFD has a higher DSR in a close-world than in open-world, which is the opposite of SAD. To our minds, a large amount of website data only appears once in open-world, even if it is confused by DFD. DSR will drop if the website will be predicted as other unmonitored website. Compared with SAD, DFD's defensive ability is not as good as SAD in open-world or close-world. When the BO increases, the gap between the DSR of the two gradually decreases.

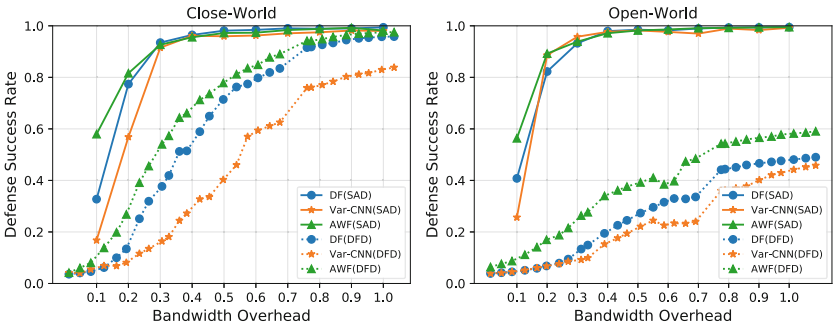

Fig. 7. Comparison of SAD and DFD in same attack model

(5) Adversary Training. Attackers can obtain defense data when the defense is public. In response to this situation, SAD can adjust BO to slow down the decline of DSR. We simulate the situation in this experiment. First, we train the DF attack model on the defense data set until ASR reaches 90%. Then, the attack model is evaluated under different BO. When the defense data sequence and the input feature dimension of the attack model are not the same, if it is not enough, it is filled with 0, and if it is too long, it will be directly truncated. The experimental results are shown in Fig. 8.

When DF uses the adversarial examples generated by SAD for training, its ASR can still reach more than 90% with the same BO. The greater the gap between BO setting of defender and attacker, the higher DSR. Therefore, SAD has excellent defense ability against WF attack model after adversary training.

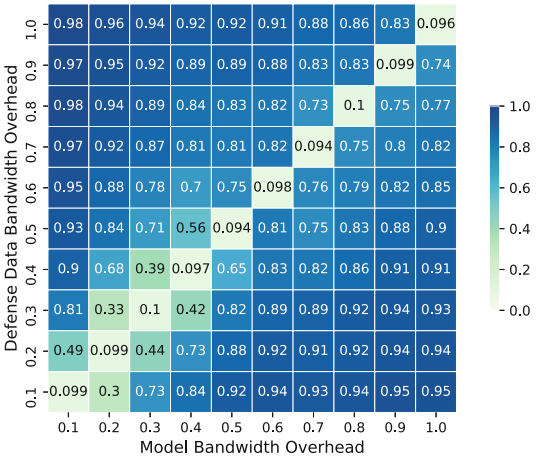

Fig. 8. Adversarial training attack models face defense data with different bandwidth overhead

(6) Black-Box Defense. The purpose of this experiment is to evaluate the robustness of SAD against unknown attack models. Unlike experiments (2)–(5), this experiment is a black-box defense. In the process of training and testing, three WF attack models can be selected, one of which is selected as the training model of SAD, and the other two is used as the test model. In Fig. 9, the dotted and solid line represents DSR under the white-box and black-box respectively. It also represents the attack model that SAD knows. Overall, the black-box defense did not cause the SAD's defense performance to fluctuate too much. No matter which attack model is chosen as the SAD training model, DSR for Var-CNN is basically the lowest. However, it is still close to saturation when the BO is 30%. In summary, SAD has good robustness facing different attack model based deep learning.

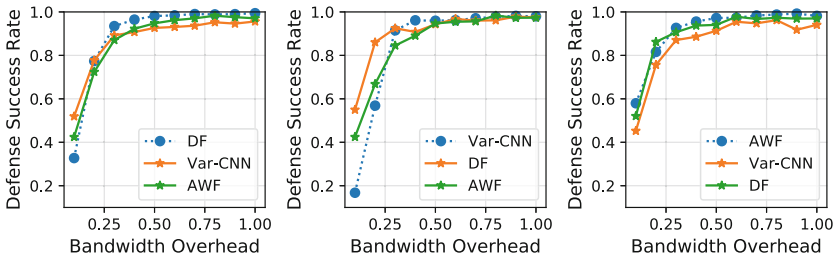

Fig. 9. SAD defense success rate in black-box defense mode

6 Conclusion

In this paper, we proposed a novel Website Fingerprinting defense, called Segment Adversarial Defense, to against deep learning-based WF attacks. SAD performs segmentation processing on live traffic, and injects dummy packets after each segment to complete the defense. SAD benefited from segmented processing can flexibly tune the bandwidth overhead. The operation effectively balance the defense success rate and bandwidth overhead. In addition, SAD can deal with adversarial training and black-box defense. Experimental results show that in a close-world and open-world, the SAD defense success rate can reach up to 99%. It only needs 30% bandwidth overhead to achieve 90% defensive success rate. In summary, SAD can effectively against WF attacks based on deep learning.

References

1. Abusnaina, A., Jang, R., Khormali, A., Nyang, D., Mohaisen, D.: DFD: adversarial learning-based approach to defend against website fingerprinting. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, pp. 2459–2468. IEEE (2020)
2. Bhat, S., Lu, D., Kwon, A., Devadas, S.: VAR-CNN: a data-efficient website fingerprinting attack based on deep learning. arXiv preprint arXiv:1802.10215 (2018)
3. Cai, X., Nithyanand, R., Johnson, R.: CS-BuFLO: a congestion sensitive website fingerprinting defense. In: Proceedings of the 13th Workshop on Privacy in the Electronic Society, pp. 121–130 (2014)
4. Cai, X., Nithyanand, R., Wang, T., Johnson, R., Goldberg, I.: A systematic approach to developing and evaluating website fingerprinting defenses. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 227–238 (2014)
5. Carlini, N., Wagner, D.: Adversarial examples are not easily detected: bypassing ten detection methods. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, pp. 3–14 (2017)
6. Dyer, K.P., Coull, S.E., Ristenpart, T., Shrimpton, T.: Peek-a-Boo, I still see you: why efficient traffic analysis countermeasures fail. In: 2012 IEEE Symposium on Security and Privacy, pp. 332–346. IEEE (2012)
7. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
8. Hayes, J., Danezis, G.: k-fingerprinting: a robust scalable website fingerprinting technique. In: 25th USENIX Security Symposium (USENIX Security 2016), pp. 1187–1203 (2016)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
10. Herrmann, D., Wendolsky, R., Federrath, H.: Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial Naïve-Bayes classifier. In: Proceedings of the 2009 ACM Workshop on Cloud Computing Security, pp. 31–42 (2009)
11. Hinton, G., et al.: Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process. Mag. **29**(6), 82–97 (2012)

12. Hou, C., Gou, G., Shi, J., Fu, P., Xiong, G.: WF-GAN: fighting back against website fingerprinting attack using adversarial learning. In: 2020 IEEE Symposium on Computers and Communications (ISCC), pp. 1–7. IEEE (2020)
13. Juarez, M., Afroz, S., Acar, G., Diaz, C., Greenstadt, R.: A critical evaluation of website fingerprinting attacks. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 263–274 (2014)
14. Juarez, M., Imani, M., Perry, M., Diaz, C., Wright, M.: Toward an efficient website fingerprinting defense. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9878, pp. 27–46. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45744-4_2
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advance Neural Information Processing Systems, vol. 25, pp. 1097–1105 (2012)
16. Panchenko, A., et al.: Website fingerprinting at internet scale. In: NDSS (2016)
17. Panchenko, A., Niessen, L., Zinnen, A., Engel, T.: Website fingerprinting in onion routing based anonymization networks. In: Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society, pp. 103–114 (2011)
18. Rahman, M.S., Imani, M., Mathews, N., Wright, M.: Mockingbird: defending against deep-learning-based website fingerprinting attacks with adversarial traces. IEEE Trans. Inf. Forensics Secur. **16**, 1594–1609 (2020)
19. Rimmer, V., Preuveneers, D., Juarez, M., Van Goethem, T., Joosen, W.: Automated website fingerprinting through deep learning. arXiv preprint arXiv:1708.06376 (2017)
20. Sirinam, P., Imani, M., Juarez, M., Wright, M.: Deep fingerprinting: undermining website fingerprinting defenses with deep learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1928–1943 (2018)
21. Szegedy, C., et al.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
22. Wang, T., Cai, X., Nithyanand, R., Johnson, R., Goldberg, I.: Effective attacks and provable defenses for website fingerprinting. In: 23rd USENIX Security Symposium (USENIX Security 2014), pp. 143–157 (2014)
23. Wang, T., Goldberg, I.: Walkie-talkie: an efficient defense against passive website fingerprinting attacks. In: 26th USENIX Security Symposium (USENIX Security 2017), pp. 1375–1390 (2017)