Generating Adversarial Packets for Website Fingerprinting Defense

Mohammad Saidur Rahman
Center for Cybersecurity
Rochester Institute of Technology
saidur.rahman@mail.rit.edu

Abstract

Website Fingerprinting (WF) is a traffic analysis attack that enables an eavesdropper to infer the victim's web activity even when encrypted and even when using the Tor anonymity system. Using deep learning classifiers, the attack can reach up to 98% accuracy. Existing WF defenses are either too expensive in terms of bandwidth and latency overheads (e.g. 2-3 times as large or slow) or ineffective against the latest attacks. In this work, we explore a novel defense based on the idea of ***adversarial examples*** *that have been shown to undermine machine learning classifiers in other domains. In our preliminary experiment, we use a simple model for generating adversarial packets using distance minimization technique that drops the accuracy of the state-of-the-art WF attack from 98% to 60%, while incurring a reasonable 47% bandwidth overhead, showing its promise as a possible defense for Tor. In this work, we have developed a novel WF defense using* ***Generative Adversarial Network (GAN)*** *that generates* ***Adversarial Packets****. Adversarial packets are padded to a Tor traffic stream in a manner that reliably fools the classifier into classifying it as coming from a different site or from a noise distribution. In our experiment, we are able to reduce the attack accuracy from 98% to 83% with significant lower bandwidth overhead which is 9%. We plan to investigate further to reduce the attack accuracy so that the defense can be realistic and deployable.*

1. Introduction

Tor is one of the most popular anonymity system with more than two millions user each day. However, Tor is known to be vulnerable to traffic analysis attacks. The adversary who observes the both entry and exit sides of the traffic in Tor is able to correlate the traffic and link the client to her destination. This type of traffic analysis attacks needs powerful adversaries. A branch of traffic analysis attacks is

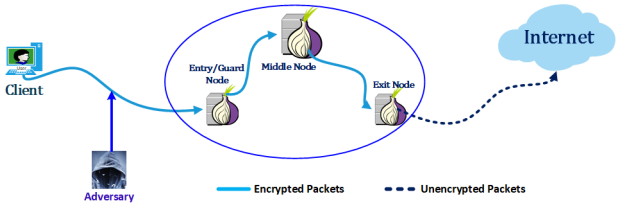


Figure 1: Website Fingerprinting Attack Model.

Website Fingerprinting (WF) attacks. The goal of the adversary in WF attacks is to identify which websites the client is visiting.

The WF attacker is considered to be a *local* and *passive* attacker (see Figure 1). The *local* means that the attacker is located in the client's network, and knows her IP, for example, she can be the client's wireless router or cable/DSL modem, the client's ISP, the guard node itself, and an AS between the client and guard node. The *passive* attacker means that the attacker only eavesdrops the traffic and does not manipulate the traffic, for example, by reshaping the traffic or tagging the traffic. Being passive makes the WF attacker almost impossible to detect. Because the WF attacker is assumed to be local and she only needs to observer the entry side of the traffic, the WF attacker is considered to be a weak adversary.

The WF attack is a supervised classification problem. The websites are the labels and the traffic traces are the instances or observations. In this work, we assume that the client is browsing the web through Tor network. The attacker uses the same privacy enhancing technology as the client, here Tor, to collect a set of instances for the websites which she is interested in identifying. From the collected instances, the attacker extracts a set of pre-defined features and trains a classifier on the extracted features. Then the attacker observes the client's traffic, extracts the features from the traffic, and classifies the traffic with the trained classifier.

The WF attacks are serious threats against Tor because the attacker only needs to observe the ingress traffic in Tor. The WF attacks have been improved over time from both feature extraction perspective and the classifier's power [7, 9, 15, 17, 21, 22]. The accuracy rate of the state-of-the-art WF attack now reaches 98% [19].

The state-off-the-art WF defense, WTF-PAD [10], injects the dummy packets to fill the gaps in the traffic and create fake bursts. Because WTF-PAD does not delay the real packets, it does not add latency overhead, and only comes at a fair bandwidth overhead cost (around 65%). WTF-PAD successfully could drop the accuracy rate of kNN [21] from 90% to 17%. A recent study [19] using a Convolutional Neural Network (CNN) could break WTF-PAD and achieve accuracy rate 90% on the traffic traces protected by WTF-PAD.

In this project, we are going to investigate a novel WF defense strategy using adversarial packets generated by a deep neural network. We plan to generate the adversarial packets using the techniques in the computer vision field such as GAN, and CycleGAN. We consider two different scenarios to evaluate the effectiveness of the adversarial examples as WF defense, one that the attacker is not aware of the defense and has been trained on the non-defended traces, and in the second scenario we consider the case that the attacker has trained the classifier on the defended traces. In our initial experiment, our new defense could drop the accuracy rate of state-of-the-art attack from 98% to 60% with 47% bandwidth overhead. Hence, we plan to investigate more to improve this defense mechanism using GAN. The purpose is to develop a realistic WF defense with acceptable bandwidth and latency overhead.

1.1. Contribution of this Paper

- Developed a novel defense using Generative Adversarial Network to *Generating Adversarial Packets*
- Evaluated Bandwidth overhead and achieved significant lower bandwidth overhead which is 9%
- Evaluated performance of Adversarial Packets against state-of-the-art WF attack lowering the attack accuracy from 98% to 83%.

2. Background & Related Work

2.1. Background

2.1.1 Tor

Tor is a low-latency anonymity network. A tor circuit consists of three nodes (i.e. guard/entry node, middle node, and exit node) (see Figure 2). Each node is aware of the addresses of the previous and the next node. For example, entry node only knows the identity of the client and the guard

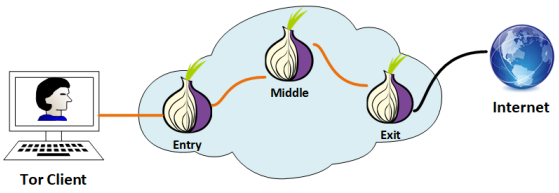


Figure 2: Tor Network.

node. Exit node only knows about the destination of the client. It uses multi-layered encryption in each node. The payloads of Tor network are encrypted using TLS. In the Figure 2, the orange line means the payloads are encrypted, and the black line means the payloads are not encrypted. An eavesdropper, who can control entry or guard node, can deanonymize a client of Tor.

2.1.2 Website Fingerprinting (WF) Attack

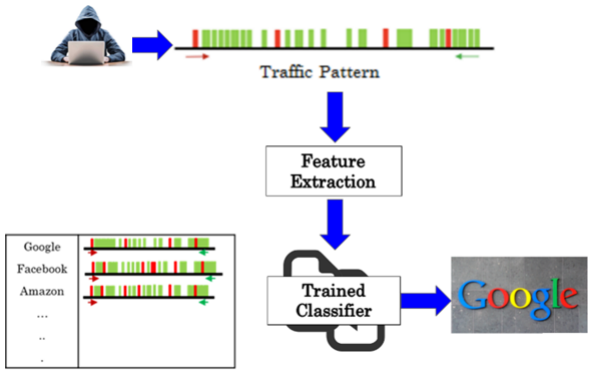


Figure 3: Feeding Traffic Pattern into Trained Machine Learning Classifier.

We can achieve privacy by separating our activity from our identity. In web browsing, our identity is our IP address and our activities are the visited websites. Tor network ensures user privacy by keeping their identity anonymous.Website fingerprinting is a type of attack that enables a local, passive eavesdropper to know the destination of a client analyzing traffic. In website fingerprinting research, We assume that a WF attacker can monitor the traffic between a client and guard node, and/or she controls the guard node.

She can collect statistical data about the traffic such as the total number of incoming packets, the total number of outgoing packets, the timing of each packet, and bursts. Packet burst is the sequence of incoming and outgoing packets. She can apply different machine learning techniques (i.e. k-nearest neighbors, support vector machines, and random decision forests) to classify websites us-

ing those traffic features and use trained machine learning classifiers to identify the clients destination websites (see Figure 3).

Website fingerprinting attacker collects traffic of several websites of her interest. These websites are called monitored websites. After that, she feeds these traffic data based on traffic features such as incoming and outgoing packets, bursts, and timestamps to machine learning model to learn from this input. This process is called training. In the next step, she feeds her collected Tor traffic to the machine learning model to predict the new output. This process is called testing

2.2. Literature Review

Tor is the most popular anonymity network to protect the user's privacy over the Internet. However, an attacker can deanonymize the activity of a Tor client by website fingerprinting (WF) attack. In 1998, WF attack was first considered as a threat against user's privacy [6]. With the growth of the Tor users over the years, WF attack has become a relevant problem for Tor. To combat this attack, Tor has implemented a WF defense [16]. In this section, we are going to discuss WF attacks and defenses in more details.

2.2.1 Attacks

In 2009, Herrmann et al. [9] published a WF attack using IP packet size for their classifier. Panchenko et al. designed a new attack adding more features: packet volume, packet direction, and timing [15]. They used support vector machines (SVM) for classification. In 2012, SVM was again used by Cai et al. who proposed a new attack based on a new representation of the classification instances [5]. Their SVM was based on the Damerau-Levenshtein edit distance and SVM kernel trick to pre-compute distance between the traces. This same attack was improved by Wang and Goldberg [22]. In 2014, Wang et al. proposed a new attack based on a k-Nearest Neighbor classifier (KNN) on a large feature set with weight adjustment [21].

Hayes et al. use yet a novel feature extraction and selection method: they use random forests to extract robust fingerprints of web pages [8]. In 2016, Panchenko et al. proposed a new attack improving features based on packet size, packet ordering, and packet direction [14]. They used the concept of Wang et al. to develop their KNN classifier based attack.

Abe and Goto is the first to explore deep learning (DL) in traffic analysis [2]. They used a *Stacked Denoising Autoencoder* (SDAE) model as their deep learning model, with a simple input data representation based on incoming and outgoing packet traces. They achieved 88% accuracy in their attack. Rimmer et al. [17] investigated three deep learning models: *Stacked Denoising Autoencoder (SDAE)*, *Convolutional Neural Network (CNN)*, and *Long-Short Term Memory (LSTM)* in WF attack. Their mainly focused on automated feature enginnering in WF attack. They also collected large datasets suitable for deep learning models, with 900 websites and 2500 traces for each site. They achieved 96% accuracy in a closed-world setting.

Sirinam et al. [19] extensively investigated the use of deep learning in website fingerprinting. Their attack could outperform all the previous attack accuracy reaching over 98% attack accuracy. They evaluated their deep learning model by 100 classes containing 1000 traces each. They developed a powerful convolutional neural network (CNN) model for their attack. Their attack achieved up to 90% accuracy against *WTF-PAD*, one of the state-of-the-art defenses. They showed the effectiveness of their attack even though the defense is in place. Their attack also achieved 49% accuracy against *Walkie-Talkie* defense.

2.2.2 Defenses

In response to the threat of WF attacks, there have been proposed several defenses against WF attacks on Tor [3, 4, 12, 13, 16, 24]. The WF defenses try to change the pattern of the traffic in a way that confounds the classifier. The change in the pattern of the traffic can happen by the link padding (the packet padding is already implemented in Tor as Tor cells are padded to 512 bytes). In the link padding strategy, dummy packets are sent to change the pattern. This type of defense is known to be too expensive in terms of bandwidth usage and latency. The super-sequence family defenses [21, 13, 24, 11] cluster the traffic into few sets. All the traffic traces in the same set are padded to the minimum sequence that contains all the sequences on that set. Beside the bandwidth and latency overhead, the problem with these defenses is that they need a large database of web-page templates for building the sets. Maintaining this database and distributing it in the Tor network is very challenging.

For the defenses against WF attacks, we are interested in the newest defenses that are claimed to be highly efficient and have low overheads in computational, storage, and bandwidth for the users.

One of the state-of-the art defenses is *Website Traffic Fingerprinting Protection with Adaptive Defense* (WTF-PAD) [10]. WTF-PAD is an updated version of of adaptive padding [18]. Adaptive padding works by sending data packets with padding of a certain distribution and no delay. By doing so, adaptive padding does not incur any latency overhead. However, it does incur a moderate bandwidth overhead. WTF-PAD resolves the issue of bandwidth overhead by applying a better and flexible distribution for padding strategies to adaptive padding. By doing so, this defense manages to achieve zero latency overhead and less than 60% bandwidth overhead.

Walkie-Talkie (W-T) is another state-of-the-art defenses developed by Wang and Goldberg [23]. The major concept of this defense is communication and burst molding. The purpose of this defense is to make two websites look exactly the same to the attacker. Their defense has 31% bandwidth overhead and 34% latency overhead.

The defense that is implemented on Tor [16] uses HTTP pipelining. This defense is implemented in response to the attack developed by Panchenko et al. [15]. The purpose of this defense is to change the order of the requests when the number of requests exceeds the depth of the pipeline. This objective is achieved by randomizing the maximum number of requests in a pipeline. The bandwidth overhead of this defense is zero. Recently, Tor has updated this defense mechanism [16] because of the emergence of newly developed strong attacks. However, neither versions of Tor's defense could reduce the attack accuracy of newly developed attacks [5, 22, 21, 2, 17, 19].

3. Generating Adversarial Traces

3.1. Dataset

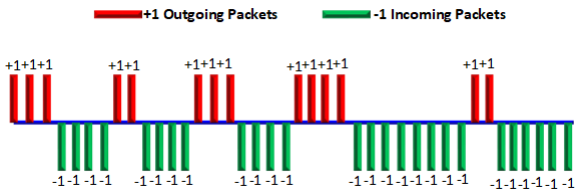


Figure 4: A Visualization of Burst of Traffic.

We apply our algorithm to generate adversarial packets on burst level characteristics as WF attack heavily relies on burst. Burst is the sequence of packets in a single direction (see Figure 4). We need to collect the traffic on the half-duplex communication. Walkie-Talkie (WT) [23] also works on the half-duplex communication and it finds the supersequence in the burst level. Sirinam el al. [19] collected a big dataset of traffic traces over the half-duplex mode. Their dataset contains 100 sites, top 100 sites in Alexa.com [1], with 900 instances for each class. This data collected in half-duplex mode over Tor network. For our evaluation, we use their dataset. We cleaned their data and removed the traces shorter than 50 packets, and the ones that their first packets are incoming packets. After the cleaning process we ended up with 83 classes with 720 instances per class. We break the data into two non-overlapping sets: the Attacker Set and the Defender Set. Each set has a *monitored set* of 83 classes, each representing a website of interest to the attacker, with 360 instances each.

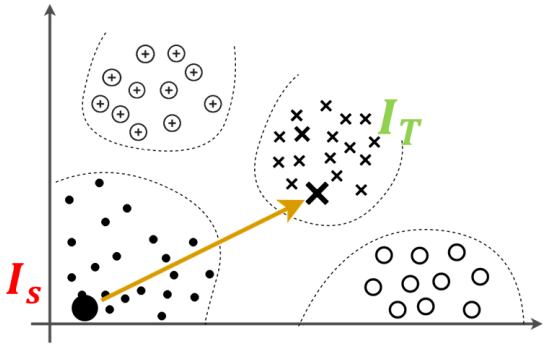


Figure 5: Adversarial Traces with Distance Minimization.

3.2. Adversarial Traces with Distance Minimization

Our current mechanism to perturb the traffic is based on *targeted adversarial examples* [20]. For a given trace, the model picks a target trace sample. The idea is to make the given trace looks like the target trace. To achieve that purpose, the model tries to minimize the distance between these two traces. Finally, the sample trace moves enough so that the classifier fails to classify the trace as coming from the right class.

More concretely, assume that we have a set of sensitive sites $\mathcal{S}$ that we want to protect and a model $f(x)$ (called *detector*) that is trained on a set of data from $\mathcal{S}$. We consider traffic trace I_s as an instance of source class $s \in \mathcal{S}$ that we want to alter such that it is classified to target class t, $t = f(I_s)$ and $t \neq s$. I_s is a sequence of the bursts, $I_s = \left[b_0^I, b_1^I, ..., b_n^I\right]$. The only allowed operation on a burst, b_i^I, is to add some positive values, $\delta_i >= 0$, to that burst, $b_i^I = b_i^I + \delta_i$. The reason for using $\delta_i >= 0$ is that we want to increase the volume of the bursts by sending dummy packets. If $\delta_i < 0$, it means that we should drop some packets to reduce the burst volume, but dropping real packets means losing data.

To protect source sample I_s, we pick p random samples from other classes, $P_{I_s} = \left[{I^0}_{T_0}, {I^1}_{T_1}, ..., {I^p}_{T_m}, \right]$. P_{I_s} is the *target pool* for I_s. ${I^j}_{T_i}$ is the j-th sample in the target pool and belongs to target class $T_i \neq s$. We want to pick a target class and re-cast the source sample to be classified as that target class. To decrease amount of change to the source sample, since adding padding adds bandwidth overhead, we pick the sample from the target pool that is closest to the source sample (see Figure 5). We define closeness using the l_2 norm distance. Formally:

$$D(x, y) = l_2(x - y)$$
$$I_T = \underset{I_t \in P_{I_s}}{\text{argmin}} D(I_s, I_t)$$

Then we modify the source sample to move toward this target sample.

Our goal is to increase the volumes of selected bursts in the source sample such that the source sample is not classified as class s and the amount of change is as small as possible to minimize the bandwidth overhead. To make the source sample to leave the source class, we move toward the nearest sample (I_T). We define Δ as the perturbation vector that we will add to the source sample to generate its defended form I_s^{new}.

$$\Delta = [\delta_0, \delta_1, \cdots, \delta_n] \quad (\forall i \in [0, \cdots, n] : \delta_i \geqslant 0)$$
$$I_s^{new} = I_s + \Delta$$

To find Δ that minimizes overhead, we should minimize distance $D(I_s^{new}, I_T)$. To do this, we compute the gradient of the distance with respect to the input. The gradient points in the direction of steepest ascent, which would maximize the distance. Therefore, we compute the gradient of the negative of the distance with respect to the input, and we move the source sample that direction towards the target sample. In particular:

$$\nabla(-D(I, I_T)) = -\frac{\partial D(I, I_T)}{\partial I} = \left[-\frac{\partial D(I, I_T)}{\partial b_i}\right]_{i \in 0, \cdots, n},$$

where b_i is the i-th burst in input I. To modify the source sample, we change bursts such that their corresponding values in $(-D(I, I_T))$ are positive. Our perturbation vector Δ is:

$$\Delta = \begin{cases} -\alpha \times \frac{\partial D(I, I_T)}{\partial b_i} & -\frac{\partial D(I, I_T)}{\partial b_i} > 0 \\ 0 & -\frac{\partial D(I, I_T)}{\partial b_i} \leqslant 0 \end{cases}$$

where α is paramter that amplifies the output of the gradient. The choice of α has an impact on the convergence and the bandwidth overhead. If we pick large value for α, we will take bigger steps toward the target sample and we will add more overhead. We modify the source sample by summing it with Δ, ($I_s^{new} = I_s + \Delta$). We iterate this process, computing Δ for each I_s and updating the source sample until we leave the source class, $f(I_s^{new}) \neq s$ or the number of iterations passes the maximum allowed iterations. In our initial experiments, we set this maximum as 200 iterations.

Because we only increase the bursts where $-\frac{\partial D(I, I_T)}{\partial b_i} > 0$, we may run into cases that after some iterations $\nabla(-D(I, I_T))$ does not have any positive values or all the positive values are extremely small such that they do not make any significant changes to I_s. In such cases, if $I_s^{new} - I_s$ is smaller than a threshold (we use threshold 0.001) for a few iterations (we use 10 iterations), and we are still in the source class, we refill the pool with new samples and pick a new target sample I_T to continue the process.

3.3. Adversarial Traces using GAN

The purpose to design a WF defense is to make the distribution of all sites almost the same. We aim to make the data inseparable so that classifiers fail to classify those websites. This is the opposite goal of *generative adversarial networks* (GAN).

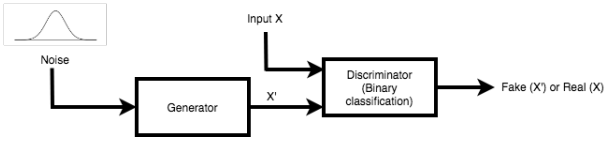


Figure 6: GAN Architecture.

GAN generates samples with many features of the real samples and the generated samples look like authentic. Two neural networks *Generator* and *Discriminator* work simultaneously in a GAN (see Figure 6). The *Generator* is a neural network that gets a randomized input drawn from a latent distribution and outputs the synthesized samples. The *Generator* should be trained such that the synthesized samples look like to be drawn from the data distribution. The *Generator* acts like a counterfeiter that generates fake notes. The *Discriminator* acts like a police, and its task is to detect whether its input is a synthesized sample or a real sample. The *Discriminator* is a neural network binary classifier. The *Generator*'s objective is to increase the error in the *Discriminator* by causing misclassification in the *Discriminator*. The objective of *Discriminator* is to discriminate the authentic samples from synthesized ones. Two networks compete with each other in *Zero-sum* game framework to reach the equilibrium point. Afterward, the *Generator* learns to map the latent distribution to the data distribution and generate the synthesized samples which look authentic.

In designing a WF defense, our goal is to map the data distribution to a latent distribution. The latent distribution can be a random noise distribution, or some target distribution. This is the opposite of the GAN's goal, GAN is mapping the latent distribution to the data distribution. Therefore, we can flip the latent and data distribution's position in GAN to reach our goal. Figure 7 shows the modified architecture of GAN for designing a WF defense.

As shown in Figure 7, the *Generator* gets the traffic traces (burst sequences),X, as the input and generates the padding (N) that should be added to the burst sequences, depending on the input traces. The *Generator* learns to adjust the padding to mimic the latent distribution. The *Generator*'s objective is to raise the error in the *Discriminator* and keep the amount of padding added to the traffic traces as low as possible to limit the bandwidth overhead. The *Discriminator* is responsible to detect whether its input drawn from the latent distribution or synthesized by the *Generator*. The latent distribution can be a normal distribution. This case is suitable for the un-targeted scenario which the defended traces are all casted to the normal distribution, their distri-

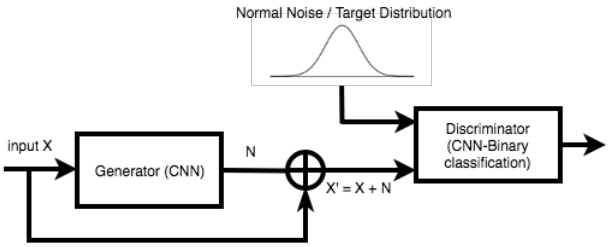


Figure 7: WF GAN Architecture.

butions are not changed to a particular target distribution. If we want to reshape the data distributions in the websites to the data distribution of a set of target websites, we can use the targets' data distributions as the latent distribution. Therefore, the *Generator* learns to change the data distribution of the sites to the target distribution.

Assuming our noise sample is N labeled as 1, and our generated adversarial sample is G_x labeled as 0. The discriminator loss is as follows:

$$L_{noise} = CrossEntropy(D(N), 1)$$
$$L_{generatedSample} = CrossEntropy(D(G_x), 0)$$

The generator loss is as follows:

$$L_{generator} = CrossEntropy(D(G_x), 1) + ||X - G_x||_2$$

4. Evaluation

In our evaluation, we break the data into two non-overlapping sets: the Attacker Set and the Defender Set. Each set has a *monitored set* of 83 classes, each representing a website of interest to the attacker, with 360 instances each. We examine the bandwidth overhead and reduction in attacker accuracy of traces protected by our method. We use traces in the training data and generated their defended forms by the method described in the previous sections. We first require a *detector* ($f(x)$) to identify when the generated samples look like a noise. Thus, we define the *detector*, a CNN model. In our evaluations we examine two cases:

- **Case I**: We train on the original traces and test on the defended traces (adversarial traces). In this case, the *detector* has been trained on the attacker set.
- **Case II**: We train on the generated traces and test on the generated traces. In this case, the *detector* has been trained on the defended samples.

We generated defended samples with various settings. We varied σ to generate the noise sample. We also varied different learning rate for the generator and the discriminator to evaluate their effect on the strength of the defended

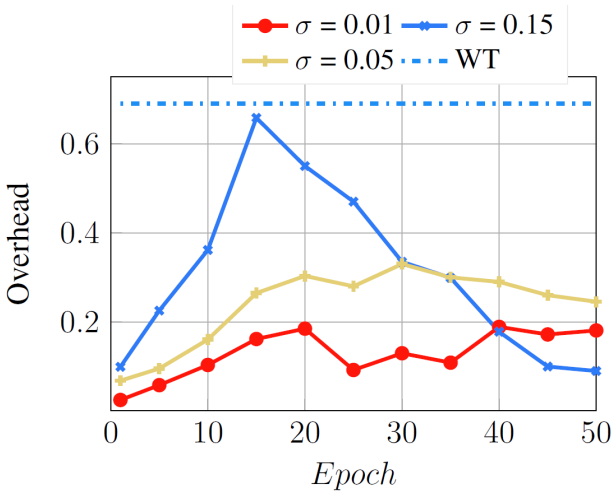


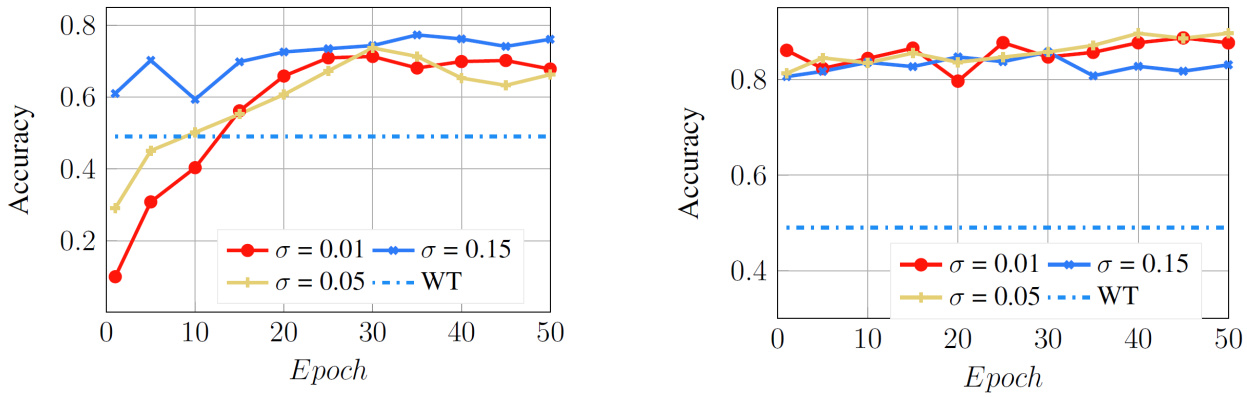
Figure 8: Bandwidth Overhead with the variation of σ for the noise distribution.

traces and the overhead. We measured the detectability of the defended samples by applying the DF attack [19] on them. Sirinam el al. [19] suggest using 5,000 packets. As both Walkie-Talkie and our method increase the size of the bursts, the number of packets in the traces increases. We thus use an input size of 10,000 packets, which is the 80th percentile of packet sequence lengths in our defended traces.

Figure 8 shows the bandwidth overhead in both Walkie-Talkie (WT) and our method as σ vary. As shown in the figure, as we vary σ, the bandwidth overhead varies. We also observe that bandwidth overhead is sensitive to the change of σ used to generate the noise distribution. The bandwidth overhead is 9% using $\sigma = 0.15$. Using $\sigma = 0.01$ and $\sigma = 0.05$, we get 18% and 24% bandwidth overhead, respectively.

Figure 9 depicts the accuracy rate of the DF attack as σ vary, for input sizes of 5,000 packets. Figure 9a and 9b depict the results of the evaluations in Case I and Case II, respectively. For **Case I**, the accuracies are 67% and 66% with $\sigma = 0.01$ and $\sigma = 0.05$. We observe increase of accuracy of **Case I** with $\sigma = 0.15$ which is 76%. For **Case II**, $\sigma = 0.15$ gives accuracy of 83% which is lower than $\sigma = 0.01$ and $\sigma = 0.05$. It is to mention that **Case II** is more realistic than **Case I** because attacker have access to the defended traffic. And it is obvious that, the attacker would train his classifier with the defended traces. According to Figure 9a, the lowest accuracy for Case I is 66% when $\sigma = 0.05$, and its corresponding bandwidth overhead is 24%. In addition, the lowest accuracy for Case II is 83% when $\sigma = 0.15$, and its corresponding bandwidth overhead is 9%.

Our evaluations show that $\sigma = 0.15$ provides lower ac-



(a) Case I

(b) Case II

Figure 9: **Accuracy**: the accuracy rate of the generated samples against the DF attack in both **Case I** and **Case II**.

curacy with lower bandwidth overhead. According to our results, our best setting is to make noise sample with higher value of σ. Though our bandwidth overhead is lower than Walkie-Talkie (WT) defense, the attack accuracy in our approach is still higher than that of Walkie-Talkie (WT) defense. In addition to that, our earlier approach to generate adversarial traces using distance minimization gave us 47% bandwidth overhead and reduced the attack accuracy from 98% t0 60%. In this new approach to generate adversarial traces using GAN, we are able to reduce the bandwidth overhead from 47% to 9%, but the attack accuracy is higher.

5. Discussion & Future Work

To make a defense effective and realistic, we have to consider the attacker having access to all the resources that a defender can use. We can see from our results that, when we train our classifier with the attacker set and test the classifier effectieness with the defended dataset, the accuracy seems better. However, when we train our classifier with the defended traces the accuracy gets higher. In the realistic setting, the attacker will get the attack accuracy similar to this. Hence, we plan to investigate further to reduce this attack accuracy. In website fingerprinting defense work, there is always a trade-off between the attack accuracy and the bandwidth overhead. Though we are able to reduce the bandwidth overhead to 9%, realistically it has no use as the attacker will get higher attack accuracy. In addition to that, this new defense approach is not doing better than the approach with distance minimization. We are also far behind to reduce the attack accuracy than Walkie-Talkie (WT) defense as well. Hence our future plan is to investigate further to modify the GAN model. We also plan to generate adversarial packets using CycleGAN. In addition, we plan to investigate different loss functions proposed in earlier work in adversarial domain.

6. Conclusion

In this work, we propose a novel defense against WF attacks with lower bandwidth overhead than Walkie-Talkie, the state-of-the-art defense, with reasonable reductions in attack accuracy. The defense uses a novel mechanism that adapts techniques leveraging generative adversarial network (GAN) used to create adversarial examples against machine learning classifiers, applying them to website traffic traces. The generated adversarial packets can limit the adversary even though he is trained on the adversarial traces. To protect a traffic trace, we add fake packets generated by the generator network to the real traffic trace and try to make the trace look like noise distribution. Our defense mechanism results in 9% bandwidth overhead and drops the accuracy rate of the state-of-the-art WF attack from 98% to 83%. We emphasize that our tests are conducted in the closed-world setting, where the attacker knows that the user is visiting one of the monitored set of websites. In the more realistic *open-world* setting, where the user could visit any site on the Web, 83% accuracy is very likely to lead to many false positives for the attacker. In future work, we plan to investigate more to improve the defense and show how to implement it.

Acknowledgment

We thank National Science Foundation for their generous grants for this project. This material is based upon work supported by the National Science Foundation under Grants Numbers 1423163, 1722743, and 1816851. We also express our gratitude to Dr. Matthew Wright and Dr. Mohsen Imani for their support, suggestions, feedback, and recommendations in this work. We also thank Dr. Christopher Kanan for his feedback in this work.

Data & Code

Data and Code will be made public upon the publication of this paper.

References

[1] Alexa. http://www.alexa.com.

[2] K. Abe and S. Goto. Fingerprinting attack on Tor anonymity using deep learning. *Proceedings of the Asia-Pacific Advanced Network*, 2016.

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

[4] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg. A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses. In *ACM Conference on Computer and Communications Security (CCS)*, 2014.

[5] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson. Touching from a Distance: Website Fingerprinting Attacks and Defenses. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.

[6] H. Cheng and R. Avnur. Traffic analysis of ssl encrypted web browsing. 1998.

[7] 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 *IEEE Symposium on Security and Privacy (S&P)*, 2012.

[8] J. Hayes and G. Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In *USENIX Security Symposium*, 2016.

[9] D. Herrmann, R. Wendolsky, and H. Federrath. Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier. In *ACM Workshop on Cloud Computing Security*. ACM, 2009.

[10] M. Juárez, M. Imani, M. Perry, C. Díaz, and M. Wright. Toward an efficient website fingerprinting defense. In *European Symposium on Research in Computer Security (ESORICS)*, 2016.

[11] L. Lu, E. Chang, and M. Chan. Website Fingerprinting and Identification Using Ordered Feature Sequences. In *European Symposium on Research in Computer Security (ESORICS)*. Springer, 2010.

[12] X. Luo, P. Zhou, E. Chan, and W. Lee. HTTPOS: Sealing Information Leaks with Browser-side Obfuscation of Encrypted Flows. In *Network & Distributed System Security Symposium (NDSS)*, 2011.

[13] R. Nithyanand, X. Cai, and R. Johnson. Glove: A Bespoke Website Fingerprinting Defense. In *Workshop on Privacy in the Electronic Society (WPES)*, 2014.

[14] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle. Website fingerprinting at internet scale. In *Network & Distributed System Security Symposium (NDSS)*, 2016.

[15] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website Fingerprinting in Onion Routing Based Anonymization Networks. In *ACM Workshop on Privacy in the Electronic Society (WPES)*. ACM, 2011.

[16] M. Perry. Experimental defense for website traffic fingerprinting. 2011. https://blog.torproject.org/blog/experimental-defense-website-traffic\-fingerprinting.

[17] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen. Automated website fingerprinting through deep learning. In *Network & Distributed System Security Symposium (NDSS)*, 2018.

[18] V. Shmatikov and M.-H. Wang. Timing analysis in low-latency mix networks: Attacks and defenses. *European Symposium on Research in Computer Security (ESORICS)*, 2006.

[19] P. Sirinam, M. Imani, M. Juarez, and M. Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning, 2018.

[20] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2013.

[21] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg. Effective Attacks and Provable Defenses for Website Fingerprinting. In *USENIX Security Symposium*, 2014.

[22] T. Wang and I. Goldberg. Improved Website Fingerprinting on Tor. In *ACM Workshop on Privacy in the Electronic Society (WPES)*. ACM, 2013.

[23] T. Wang and I. Goldberg. Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In *USENIX Security Symposium*, 2017.

[24] C. V. Wright, S. E. Coull, and F. Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *Network & Distributed System Security Symposium (NDSS)*, 2009.