Poster: Website Fingerprinting Attacks with Timing-based Features using Capsule Networks

Payap Sirinam, Mohammad Saidur Rahman, Kantha Girish Gangadhara, Matthew Wright
Center for Cybersecurity, Rochester Institute of Technology
{payap.sirinam, saidur.rahman, kantha.gangadhara}@mail.rit.edu
{matthew.wright}@rit.edu

***Abstract*—The *Tor* anonymity system helps Internet users to hide their online browsing behavior. Unfortunately, the *website fingerprinting (WF)* attack has become one of the most serious threats for *Tor* users. Previous studies have shown that WF attacks using deep learning techniques such as Convolutional Neural Networks (CNN) are very effective and can undermine the WTF-PAD defense that is being considered seriously for deployment in Tor. These attacks, however, cannot break the recently proposed Walkie-Talkie defense, which effectively conceals the packet sequences that are used as the main features for most WF classifiers. Since most attacks do not consider timing features, Walkie-Talkie does not seek to modify the timing of packets. Thus, it is important to understand the extent to which timing features can be effectively used in WF attacks and potentially undermine Walkie-Talkie. We applied the Capsule Network (CapNet) deep learning architecture to create a WF classifier with timestamps as the data input representation. Our experimental evaluation shows that using timestamps with CapNet provides promising results on unprotected Tor data with 88% accuracy and an acceptable cost for training. We believe that further refinement is possible to improve the WF attack and will explore this in future work.**

I. Introduction

Tor provides anonymity to millions of users as they visit sites and services online. Recent work has shown, however, that an eavesdropper can unmask the sites that a client is visiting by using a traffic analysis attack called *website fingerprinting* (WF). In a WF attack, depicted in Fig. 1, the adversary observes the traffic between the client and the *guard*, the first node in the user's Tor circuit. He then extracts input features from the traffic and supplies them as input to a machine learning (ML) classifier that aims to recognize which website the user has visited. Recent WF attacks have been shown to be effective with over 90% accuracy against Tor using classifiers such as k-NN [1], SVM [2] and k-FP [3].

In the past year, studies have shown that the adversary could apply deep learning techniques such as Convolutional Neural Networks (CNN) to perform even more effective WF attacks. Rimmer et al. [4] used deep learning to automatically extract traffic features for WF attacks and could achieve 96% accuracy using CNN. However, Rimmer et al. did not evaluate their attacks against existing WF defenses such as WTF-PAD [5] and Walkie-Talkie (W-T) [6]. Sirinam et al. [7] explored the design of an improved CNN classifier with that reached 98% accuracy. Moreover, they showed that their CNN WF attack attains over 90% accuracy against WTF-PAD, a defense that is currently being implemented in Tor [8].

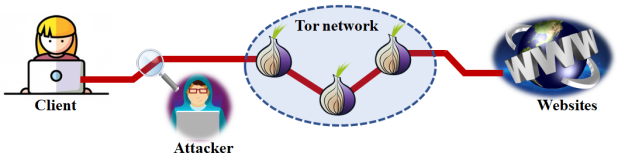

Fig. 1. The WF threat model

To date, no attacks have proven to be effective against W-T. W-T creates a super-trace, in which least two different websites' traces are molded together in terms of packet direction and size. Since the state-of-the-art WF attacks have mainly relied on the use of packet direction and size as the data input representation, W-T is very effective at undermining these attacks. As a result, the classifier cannot gain any useful information to make a prediction based upon the distinction among different packet sequences.

In this paper, we conduct a preliminary experimental evaluation to investigate the use of a different data input representation for WF attacks based on packet timing. Most attacks ignore timing information or deemphasize timing features [3], as it can be very noisy. We apply the recently proposed deep learning technique called Capsule Networks (CapNets) [9], which was shown to outperform CNNs in image recognition tasks, and we find that it offers promising initial performance on timing data. As future work, we plan to further investigate how to effectively apply timing as the additional features to help the classifier make better and more accurate predictions, especially against state-of-art defenses and in larger-scales attack in the open-world scenario.

II. WF Attack using Capsule Networks

Dataset. We use the same dataset as Sirinam et al. used for their study on deep learning [7]. The dataset contains 95 monitored websites (classes), where each website was downloaded 1000 times. Each download generates one *instance* that is represented as a sequence of tuples $<timestamp, \pm packet\_size>$, where the sign of $packet\_size$ indicates the direction of the packet: positive means outgoing and negative means incoming. We use hold-out validation to categorize dataset into three groups for each website with a ratio of 8:1:1 for training:validation:testing instances.

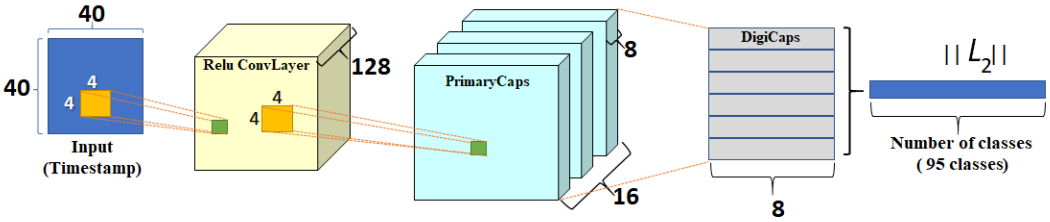

Fig. 2. Hyperparameters in our CapNet model

TABLE I
TIME REQUIRED TO TRAIN THE MODEL FOR DIFFERENT TRAINING SIZES

Training Size / Class	80	240	400	560	640	800
Training Time (Hours)	4.4	14.1	23.3	32.7	37.1	45.3

Hyperparameter Tuning. Compared to CNNs, CapNets require significantly more resources such as the number of parameters and longer training time. To apply CapNet with our large dataset, we have to perform hyperparameter tuning to create a model that can be run with our limited computing resources (NVIDIA Quadro P5000 with 16 GB of GPU Memory) and still provide good performance. Fig. 2 visualizes the selected hyperparameters used in our model. In future research with more computing resources, the model will be able to be optimized further and achieve better performance.

Data input. In this work, we mainly focus on the use of timing information as the new data input representation. Thus, we only extract the timestamp from each sequence of tuples to create a list of timestamps. We then adjust the data input by following the CapNet implementation. In the CapNet, the sequence of timestamps is reshaped into a 2D matrix. We studied the appropriate shape for the input and found that 40 rows x 40 columns provides reasonable performance and still allows the CapNet runnable with our limited resources.

Results. We implemented the CapNet by using Keras with Tensorflow as the backend. We first evaluate the effect of training input size on its accuracy as shown in Fig. 3. We found that the accuracy gradually increased as with more training inputs, slowly improving up to 720 training inputs and then began leveling off. The maximum accuracy was 88%, which suggests that using packet timing is a promising research direction to improve the performance of WF attacks.

We also investigate the training time, the total amount of time required to complete training the WF classifier using CapNet. The time required for training is an important factor for the practical implementation of the model. Table I shows that the rate of training time required to create the CapNet WF model grows linearly with the number of training inputs. The most accurate model required approximately two days to create the WF classifier, which is much slower than we found for CNN (about two hours), but may be acceptable for a well-resourced attacker.

III. Conclusion and Future Works

In this paper, we investigate the use of packet timing as a feature to perform website fingerprinting (WF) attacks.

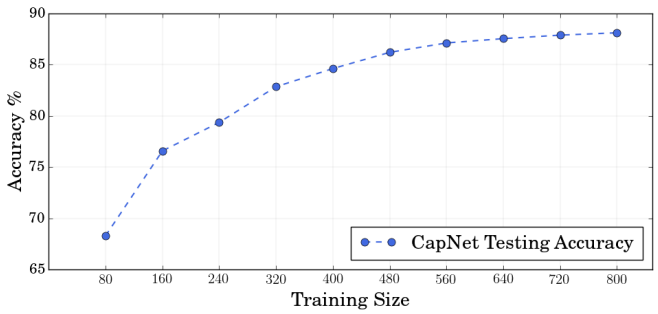

Fig. 3. Accuracy vs. training size

Using a CapNet classifier and optimizing the hyperparameters, we attained up to 88% accuracy with an acceptable cost of training. For future work, we plan to further investigate and improve the CapNet architecture to improve the performance of the attack. Moreover, we will study the use of packet timing in the more realistic open-world scenario and against defenses such as WTF-PAD and Walkie-Talkie.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grants No.1619067.

References

[1] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *USENIX Security Symposium*, 2014.
[2] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel, "Website fingerprinting at Internet scale," in *Network & Distributed System Security Symposium (NDSS)*, 2016.
[3] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *USENIX Security Symposium*, 2016.
[4] V. Rimmer, D. Preuveneers, M. Juarez, T. V. Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," in *Network & Distributed System Security Symposium (NDSS)*, 2018.
[5] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an efficient website fingerprinting defense," in *European Symposium on Research in Computer Security (ESORICS)*, 2016.
[6] T. Wang and I. Goldberg, "Walkie-talkie: An efficient defense against passive website fingerprinting attacks," in *USENIX Security Symposium*, 2017.
[7] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," *arXiv preprint arXiv:1801.02265*, 2018.
[8] M. Perry, "Padding negotiation." Tor Protocol Specification Proposal. https://gitweb.torproject.org/torspec.git/tree/proposals/254-padding-negotiation.txt, 2015. (accessed: October 1, 2017).
[9] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems (NIPS)*, 2017.