2015 IEEE Symposium Series on Computational Intelligence

P2V: Effective Website Fingerprinting Using Vector Space Representations

Khaled Al-Naami, Gbadebo Ayoade, Asim Siddiqui, Nicholas Ruozzi, Latifur Khan and Bhavani Thuraisingham
Computer Science Department, The University of Texas at Dallas
{khaled.al-naami, gbadebo.ayoade, asim.siddiqui, nicholas.ruozzi, lkhan, bhavani.thuraisingham@utdallas.edu}

***Abstract*—Language vector space models (VSMs) have recently proven to be effective across a variety of tasks. In VSMs, each word in a corpus is represented as a real-valued vector. These vectors can be used as features in many applications in machine learning and natural language processing. In this paper, we study the effect of vector space representations in cyber security. In particular, we consider a passive traffic analysis attack (Website Fingerprinting) that threatens users' navigation privacy on the web. By using anonymous communication, Internet users (such as online activists) may wish to hide the destination of web pages they access for different reasons such as avoiding tyrannical governments. Traditional website fingerprinting studies collect packets from the users' network and extract features that are used by machine learning techniques to reveal the destination of certain web pages. In this work, we propose the packet to vector (P2V) approach where we model website fingerprinting attack using word vector representations. We show how the suggested model outperforms previous website fingerprinting works.**

I. Introduction

Technological advances have resulted in tremendous growth in information access with the expansion of the world wide web. Recently, web navigation privacy has gained increasing interest from both attackers seeking identification and defenders seeking anonymity [1]. The attacker such as a repressive government tries to reveal web page destinations of online activists, journalists, or bloggers in order to censor. This is known as a Website Fingerprinting (WF) attack. The defender such as Tor anonymity network [2] uses defenses to try to fool the attacker by changing the characteristics of the network traffic.

To protect user's privacy, various techniques such as SSL, IPSec, and VPN have been developed by defenders to hide the web access traffic [3]. These privacy protection techniques are particularly valuable for users who may wish to be anonymous especially in countries where strict censorship on the Internet is imposed. This has led to attackers (or adversaries) utilizing all the means of traffic analysis to identify the web pages visited by the Internet users.

The competition between attackers and defenders is continually evolving. On one hand, the attacker collects the packets exchanged between the client and server, extracting patterns and features and performs various traffic analysis through statistical or machine learning methods (e.g., [4], [5], [6], [7], [8]) and attempts to predict the particular website an Internet user is trying to access. On the other hand, defenders have been developing various means to thwart such attempts by disguising and encrypting network packets bound for a particular destination.

The fundamental concept of website fingerprinting is that every website can be assumed to have unique content. The content of a web page is typically downloaded to a client browser as a sequence of multiple packets, depending on the network protocol. These packets (assumed encrypted) exhibit various structural properties, such as packet length, that can be used to identify the website.

In semantic vector space models (VSMs) of language, each word is represented as a real-valued vector. These vectors can be utilized as features in a variety of natural language processing and machine learning tasks [9], [10], [11]. The constructed word vectors exhibit interesting semantic and syntactic regularities. For example, in word vector space, the sentence "king to queen is as man to woman" was proven to be represented as king $-$ queen $=$ man $-$ woman [11]. There are many proposed methods for VSMs. They take a text corpus as input and give us word vectors. Initially, a vocabulary or dictionary is built from the training data and then based on the followed approach, the vectors are learned for each word. Mikolov et al. [12] introduced a word to vector algorithm that focuses on representations of words learned by neural networks. Most recently, Pennington et al. [9] proposed "GloVe", a global vector log-bilinear regression model that uses global matrix factorization and local context window methods.

Existing website fingerprinting studies focus on collecting packets from the user's network and extract statistical features which are used by machine learning techniques to predict the destination of web pages. In this paper, we propose the packet to vector (*P2V*) approach. We model the website fingerprinting attack using the Global Vector space representation (GloVe) [9] as one of the most recent word vectors methods. We construct a corpus from network packets and represent these packets as real-valued vectors. We show how global log-bilinear regression models are appropriate to improve the website fingerprinting attack. We demonstrate how the suggested model outperforms previous website fingerprinting works.

The intuition behind P2V is as follows. Communication between client and server is stacked over the TCP protocol. Based on connection measures like network congestion, the TCP protocol uses flow control mechanisms such as window size and scaling, acknowledgement and sequence numbers, and others to ensure a certain understanding between client and server. Accordingly, the number of bytes (packet lengths) and hence time of transmitted packets in each direction are decided based on this fact. This means each packet flow affects the subsequent packet flow. This mechanism continues until communicating parties flag to finalize the connection. We view this understanding between client and server as a language or dialogue between two parties.

Moreover, the GloVe model leverages statistical information by training on elements in a word-word co-occurrence matrix. In this matrix, based on a sliding context window, each element X_{ij} tabulates the number of times word i occurs in the context of word j. As described above, in TCP, each packet flow affects the next packet flow. This means there is a dependence between consecutive packet flows in a TCP connection. Hence, we build a packet-packet co-occurrence matrix which gives us meaningful counts for each trace (or website download).

Previous website fingerprinting studies ignored the TCP flow control packets (like the ACK packets) as they decrease accuracy and do not provide distinguishing statistical benefits between websites. In this paper, it is the first time that the TCP flow control packets are utilized. We show how the ACK packets are essential to build the corpus as they enrich the vocabulary. Compared to NLP, the ACK packets may act as filter or stop

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.19

words in English. Although some applications tend to eliminate these words as they are considered noise, there are other domains such as Author Attribution where these words are considered important as they provide distinguishing writing styles [13]. In website fingerprinting, the ACK packets are important as they provide accurate fingerprints for each website. In addition, as mentioned earlier, the GloVe model depends mainly on the co-occurrences of words (context-counting). It does not neglect stop words from the corpus. Instead, it assigns different weights for frequent co-occurrences.

In short, the main contributions of this paper are summarized as follows.

1) We propose a packet to vector (P2V) model for the website fingerprinting attack. We build a corpus from network packets and represent these packets as real-valued vectors.
2) Unlike previous website fingerprinting works, we consider the TCP ACK packets as essential elements to build the corpus as they enrich the vocabulary and hence increase the accuracy of the website fingerprinting attack.
3) We show how P2V can remarkably increase the accuracy of website fingerprinting when compared to previous approaches.
4) We also show that our P2V technique is more immune and resilient to website fingerprinting countermeasures (defenses) than previous classifiers.

The rest of the paper is organized as follows. In Section II, we present relevant background information and related studies. We then present our approach in Section III. The model is evaluated in Section IV. Here, we also compare our results with previous studies. The assumptions and consequences of the new attack are discussed in Section V. Finally, we conclude our paper in Section VI.

II. Background

In this section, we present relevant background. We start by explaining the Global Vector for word representation model (GloVe) [9]. We then give an overview on the website fingerprinting attacks, and finally discuss the defense mechanisms in website fingerprinting.

A. GloVe

Vector space models (VSMs) of language have proven to be very useful in many NLP and machine learning tasks. In VSMs, each word in a corpus is represented as a real-valued vector. These vectors can be used as features in many applications. Word to vector [12] and GloVe [9] are two of the most recent algorithms used for building word vectors. Mikolov et al. [12] presented a word to vector algorithm that uses neural networks to learn representations of words. Most recently, Pennington et al. [9] proposed Global Vectors for Word Representations (GloVe in short). GloVe was shown by authors to outperform many VSMs including the word to vector [12] method mentioned above. Hence, in this paper, we use GloVe as one of the newest methods, to model the website fingerprinting attack.

Given a text corpus as input, GloVe builds word vectors in an unsupervised learning manner. The basic idea is to use word statistics as the primary source of information by examining word co-occurrences in the corpus. In a high level overview, we can summarize the GloVe algorithm as follows. Before training the model, we first construct the word-word co-occurrences matrix. Then considering word pairs, GloVe finds a log-bilinear regression model that includes word vectors as parameters. Finally, using any gradient descent algorithm, the model parameters (word vectors) are computed. We now present the GloVe algorithm in more details.

- ***The Matrix***. GloVe starts off by running through the corpus once to build the global word-word co-occurrence matrix X. Based on a sliding context window, each entry X_{ij} tabulates the number of times word i occurs in the context of word j. The result is a sparse matrix with a lot of zero entries. If the corpus is large, this counting step may be expensive. However, it is just a single pass that happens only once. The advantage about GloVe is that it trains the model on the non-zero entries of X which makes the training iterations much faster. Now that X is ready, we will use it in place of our corpus.

- ***The Model***. Generally speaking, given a sample data on two variables x and y, the equation $y = \beta_1 x + \beta_0$ is considered one of the simplest linear models, where β_1 is the slope and β_0 represents the intercept with the y-axis. Learning the optimal parameters β_0 and β_1 gives the best line (predictor) that ties variables x and y. One of the techniques used is to minimize a loss or objective function by using the gradient descent iterative algorithm.

  GloVe follows a similar approach. It constructs a model for the variable X_{ij} in the co-occurrence matrix X. The model has parameters (word vectors) to be learned by minimizing an objective function using a gradient descent-like algorithm. GloVe's concept revolves around the notion that word vector spaces have substructure that should be considered when designing algorithms to build word vectors. Typically, for nearest neighbor tasks, the existing similarity metrics such as Euclidean distance (or Cosine similarity) produce a single scalar value that may not capture intricate relationships between words. The GloVe model suggests using the vector difference between the two word vectors as this captures more interesting and useful meanings. The word vector learning model has been built considering the ratios of co-occurrence probabilities between words which can be calculated directly from X. The result is the log-bilinear regression model in Eqn. 1 for each word pair of word i and word j.

$$w_i^T w_j + b_i + b_j = \log X_{ij}, \tag{1}$$

  where the d-dimensional word vectors $w_i, w_j \in \mathbb{R}^d$ and b_i, b_j are scalar bias terms associated with words i and j. The model in Eqn. 1 constructs word vectors that are guaranteed to retain useful information about co-occurrence of words i and j.

- ***The Objective Function***. To learn the parameters w_i, w_j, b_i, and b_j, we need to minimize an objective or cost function that considers the model in Eqn. 1. However, the problem with this model is that it produces equal weights for all word-word co-occurrences in X. As some words may co-occur rarely, their co-occurrences are noisy and can be neglected. To eliminates the noise effect, the following weighted least squares model is introduced.

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij}) \ (w_i^T w_j + b_i + b_j - \log X_{ij})^2, \tag{2}$$

  where V is the vocabulary size and $f(X_{ij})$ is a weighting

function to eliminate noise. In order for $f(X_{ij})$ to work as such, it should satisfy some properties. It should be non-decreasing to deal with rare co-occurrences. Also, for large values of X_{ij}, $f(X_{ij})$ should return 1 so frequent co-occurrences are not overweighted. The following equation shows how $f(X_{ij})$ satisfies the properties mentioned above.

$$f(X_{ij}) = \begin{cases} (\frac{X_{ij}}{x_{max}})^{\alpha}, & \text{if } X_{ij} < x_{max} \\ 1, & \text{otherwise.} \end{cases} \quad (3)$$

The weighting function simply returns 1 if $X_{ij} \geq x_{max}$. For all other co-occurrences, a value between 0 and 1, controlled by α, will be returned. The authors have found the model performs best with $x_{max} = 100$ and $\alpha = 3/4$.

Running gradient descent on Eqn. 2 learns the values of the word vectors which can be used as features in many NLP and machine learning tasks.

B. Website Fingerprinting

Typically, the content of web pages are protected by encrypting network packets in transit, and destination IP addresses are hidden using proxy servers. Due to uncertainty and noise in data, machine learning algorithms are used to learn a probabilistic model that can capture various characteristics of a website from encrypted network packets. In this paper, we use the term "website" and "webpage" interchangeably. Website fingerprinting is essentially the traffic analysis of network packets, exchanged between a client browser and a server to load the main page of a website on the client's browser, with the goal of predicting the website destination.

Encryption methods such as SSL are used over HTTP to protect web content against traffic surveillance, to form HTTPS. However, studies [14], [15] have found that such encryption methods are insufficient to protect user identity from traffic analysis. Web proxies are used, along with data encryption, to form an anonymous network. The Tor anonymity network [2] is developed on this notion, to hide the information and activities of its users by providing a low latency and pipeline randomization to counter passive traffic analysis. A circuit of three relay nodes is formed within the Tor network consisting of an entry node, exit node, and a randomly selected relay node. Circuit connections are reestablished approximately after every 10 minutes of usage [16]. Figure 1 depicts an example of a client (desktop browser or mobile device) connecting to a server using the Tor network.

1) Classification: Various traffic analysis techniques [5], [17] have been proposed to perform website fingerprinting on the Tor network. Since the primary problem in website fingerprinting is to determine the website accessed by a user, this can be treated as a classification problem where a class label is the website name. A network trace is a sequence of packets exchanged by the server and client in order to load a webpage on the client's browser. A classifier is built using traces from multiple websites. The information present from network packets in each trace is typically summarized to form a histogram feature vector, where the features [4] include packet length and direction with respect to the client browser. In addition to using packet length histograms, Panchenko et al. [18] introduced *Size Markers or Bursts*. A burst is a sequence of consecutive packet flows in the same direction. Burst size is the summation of its packet lengths. A burst histogram is then built from the bursts of the trace. Panchenko used other features such as unique packet sizes, HTML markers, and percentage of incoming and outgoing packets. Dyer et al. [17] used bandwidth, website upload time, and bursts as features.

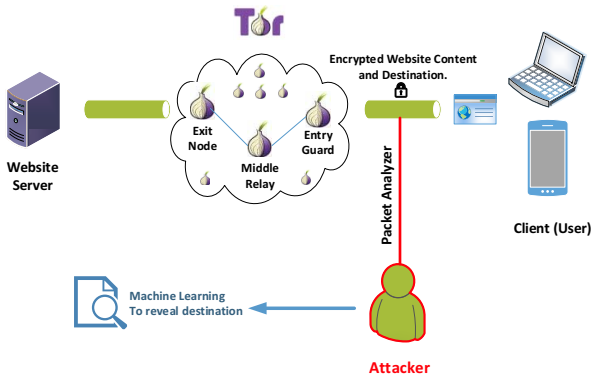


Fig. 1: An example of Tor. A client or user connects to the Internet (server) using Tor network. The three Tor nodes are shown. The website fingerprinting attack occurs between the user and the Tor entry guard.

Wang and Goldberg [19] used Tor cell traces to improve website fingerprinting on Tor.

The classification techniques used in these studies include Naive Bayes, SVM, Decision trees and K-NN. These algorithms are trained using training data, and are used for predicting the class label of a test trace. In order to train a classification model (or a classifier), sufficient number of traces from every website is required. Due to the existence of a large number of web pages in the world, two scenarios are typically considered for classification. These are called open-world and closed-world [7], [18]. A Closed-world attack assumes a finite number of websites. Traces are collected from these websites for training. For testing, the client is assumed to access these websites only. Multi-class classification is then used. In an Open-world scenario, the attacker monitors a small set of websites. These are called "monitored" websites. Traces are collected from monitored and non-monitored websites for training. The prediction becomes a binary classification problem. This paper focuses on the closed-world scenario.

C. Defenses

The topic of website fingerprinting defense has been an active area of research. Several defenses have been proposed to resist website fingerprinting attacks. All of the defenses aim to obfuscate the pattern of the packets of the loaded website. These defenses vary from padding packets with extra bytes to morphing the website packet length distribution and make it appear to come from another target distribution (i.e., a different website) [17]. In packet padding, each packet size in the trace is increased to a certain value depending on the padding method used.

Padding techniques come with the overhead of appending large number of bytes to packets. Therefore, several other smart padding defenses have been introduced. We describe three of the most effective website fingerprinting defenses that we consider when evaluating our approach later when we discuss results.

- Pad to MTU. MTU (Maximum Transmission Unit) determines the maximum size of each packet in a communication between two ends. In a Pad to MTU defense [17], each packet is padded to the maximum size (MTU). This technique prevents the attacker from extracting

detailed packet lengths distribution information which help machine learning classifiers to identify webpages. There is a tradeoff, however, when using this defense as it comes with a high cost of appending bytes to every packet of size less than MTU.

- **Direct Target Sampling (DTS). DTS was proposed by [20]. It is considered as a distribution-based defense which makes the packet length distribution of a certain website appear as coming from a different website distribution. It has an advantage over the pre-packet padding techniques, like Pad to MTU, in that it requires less overhead by appending less bytes depending on the distribution of the target webpage. Furthermore, as a distribution-based technique, DTS defense proves to be more effective than the traditional packet padding.**

  As an example, lets consider two webpages S and T where S is the source and T is the target. We derive two distributions D_S and D_T from their packet length histograms. From D_T probabilities, we build the target Cumulative Distribution Function (CDF_T) to sample random variables (packet lengths) for each packet in S by running a pseudorandom number generator to get a number between zero and one inclusive. So for every P_i (packet of length i in S) , we sample P_j (a packet of length j using CDF_T). If $j > i$, we pad P_i to length j and send, otherwise, we send P_i as is. We continue random sampling from D_T until all S packets have been consumed. The result is a new distribution D_N.

  In addition, we continue sampling from D_T until the $L1$ distance between the new distribution D_N and the target distribution D_T is less than a predefined threshold, which empirically was determined to be 0.3 [20].

- **Traffic Morphing (TM). TM [20] is similar to DTS but reduces the cost or overhead by using Convex Optimization methods. Wright et al. [20] introduced the cost function as the objective function that we like to minimize. The convex optimization parameters are probabilities in an $m \times m$ two dimensional array A where m is the MTU in TCP/IP transmission.**

  Figure 2 depicts this process. Each column in A is a Probability Mass Function (PMF) whose values sum up to 1. Similar to DTS, from each column's PMF, we generate the corresponding CDF. We do that in DTS, but we do it once for the target website distribution D_T.

  As an example, to morph the source website S to the target website T, we need to learn the parameters in A such that $T = AS$. For each packet of length i in S, P_i, we go to the i^{th} column in A and run a pseudorandom number generator over its cumulative distribution function (CDF_i) to sample P_j (a packet of length j from T). If $j > i$, then we pad P_i to length j, otherwise, we split packet P_i and send. Wright et al. [20] introduced other constraints in the convex optimization method as $i < j$ so there is no need to split packets from the source website as this affects the quality of some applications like streaming data as in audio or video. The result is a new distribution D_N.

  Similar to DTS, we continue sampling from D_T until the L1 distance between D_N and the target distribution D_T is less than 0.3.

There are other defenses that have been proposed. Authors in [17] introduced another defense that not only incurs a packet size overhead, but also requires a packet transmission delay. The defense has been experimentally proven to be inefficient by the authors in their paper. Cai et al. [8] improved the defense in [17] by optimizing size and delay parameters. Trying to change packets sending and arrival times by delaying transmission may not be practical as this increases network latency. Furthermore, TCP packet retransmission may be a result of packet delay. According to RFC 2988 [21], TCP retransmission occurs if the RTO (Retransmission Timeout) is exceeded.

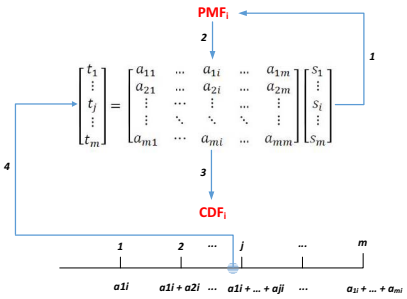


Fig. 2: Traffic Morphing.

III. P2V Approach

In this section, we present the details of our Packet to Vector (P2V) approach and explain how we utilize word vector representations to improve the website fingerprinting attack.

A. Concept

Previous studies [7], [15], [17], [18] on website fingerprinting used features such as time taken to load the webpage, packet size with direction of data transmission, packet order, and the length of combined sequential packets in the same direction, called burst (for instance, see Uplink burst in Figure 3). These features are extracted from a trace of network traffic belonging to a single website. New features are then created by bucketizing the transmission length and counting the frequencies within each bucket [18]. Therefore, each trace of a webpage would have a large number of features. If m is the total number of features, then each trace can be seen as an m-dimensional vector. We will see how this dimensionality issue is to be solved by the P2V model where a low d-dimensional vector is produced and used for classification. A class label (i.e. webpage name) is assigned to this trace.

In this work, we take a new packet to vector (P2V) approach to improve this attack. We show how we model website fingerprinting using word vector representations. More specifically, we use the GloVe model described in Section II-A. The GloVe model uses the context-counting model which leverages statistical counts by training on elements in a word-word co-occurrence matrix. Each element in this matrix tabulates the number of times word i co-occurs in the context of word j. In the TCP protocol, each packet transmission affects the following packet transmission. TCP uses flow control mechanisms such as window size and scaling, ACK packets and other methods to ensure safe arrivals of packets in both directions. This means there is a dependence between consecutive packet flows in a TCP connection. We will shortly explain how we construct the corpus

by going through packets in sequence in order to build a packet-packet co-occurrence matrix which guarantees to give us meaningful counts for each trace.

B. PORDs

The GloVe model takes a text corpus as input and produces a vector for each word in the vocabulary. In website fingerprinting, as depicted in Figure 3, all we have is just trace packets collected while downloading websites. We need a mechanism to translate these packets into text tokens. We call these tokens *PORDs* (for Packet wORDs). PORDs are extracted from the sequences of packets. Unlike previous studies on website fingerprinting which ignored the ACK packets as they were considered noise, our P2V model does use the ACK packets to generate PORDs. In the evaluation section, we show how the ACK packets enrich the vocabulary and produce better results. For ease of analysis, we organize generating PORDs into the following categories.

1) Packet Length PORDs: **For each packet, we take the packet length in bytes and construct the packet PORDs. We consider both uplink and downlink directions. An uplink (or *Tx*) packet PORD with a length l is different than a downlink (or *Rx*) packet PORD with the same length l.**

2) Uni-Burst Size PORDs: **Burst (or Uni-Burst) consists of consecutive packets in the same direction. As illustrated in Figure 3, we call the burst going from user to a server an uplink or *Tx burst* and the burst coming from a server to user a downlink or *Rx burst*. Burst size is the summation of all of its packet sizes. We take each uni-burst size as a PORD. We also consider the direction in this category as well. We bucketize as this gives us best results as to be shown in the evaluation.**

3) Uni-Burst Time PORDs: **Packet time is the departure/arrival timestamp in uplink/downlink direction. This is measured at the client side by the eavesdropper (attacker). Burst time is the difference between the last packet and first packet times (the time it takes the burst packets to get transmitted/received by the client in any direction). A PORD is constructed here from each Tx/Rx uni-burst time.**

4) Uni-Burst Count PORDs: **Uni-Burst count is the total number of packets contributing in the burst. We take each Tx/Rx uni-burst count as a PORD.**

5) Bi-Burst Size PORDs: **Bi-Burst is the sequence of two adjacent bursts. As shown in Figure 3, Rx-Tx-Burst is the combination of downlink and uplink consecutive bursts. We take the two sizes of each of the two bursts as a new PORD. Direction is considered here as well. Tx-Rx-Burst size is different than Rx-Tx-Burst size. We use bucketizing as above.**

6) Bi-Burst Time PORDs: **This set of PORDs is similar to the Bi-Burst size PORDs approach described above but we take the two time differences in each of the adjacent bursts.**

Table I summarizes how we generate PORDs from packets, uni-bursts, and bi-bursts sequences.

C. POCUMENTs

As described in Section II-A, the primary source of information is the word co-occurrences matrix. Running through words (or PORDs) in the documents, we build statistical counts of the number of times any two words co-occur together in a context window. In Section III-B, we discussed how to generate PORDs. In this section, we show how to organize these PORDs in POCUMENTs (short for Packet dOCUMENTs). The juxtaposition of the PORDs in our POCUMENTs is important as GloVe captures

TABLE I: Generating PORDs from Packets, Uni-Bursts, and Bi-Bursts.

Category	No	PORDs
Packet (Tx/Rx)	1	Packet length
Uni-Burst (Tx/Rx)	2	Uni-Burst size
	3	Uni-Burst time
	4	Uni-Burst count
Bi-Burst (Tx-Rx/Rx-Tx)	5	Bi-Burst size
	6	Bi-Burst time

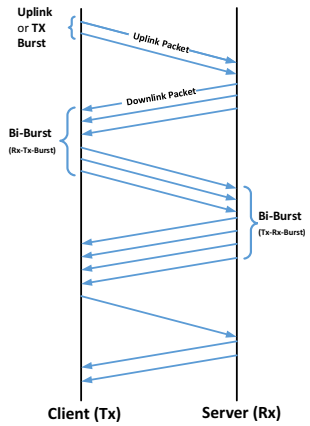


Fig. 3: Sequence Diagram between Client (Tx) and Server (Rx). Packet, uni-burst and bi-burst transmissions between two ends are illustrated.

useful statistics specified by that. Each trace (webpage load) is considered as a POCUMENT.

For a single pass in each trace used to train the model, we run through packets in order of departure/arrival from/to client side to construct the POCUMENT. For the purpose of illustration, in Figure 3, we run through packets from top to bottom and buffer all PORDs we will use. We insert the PORDs in each POCUMENT in an order described as follows (Notice that this is the best order after trying multiple combinations).

- **We first insert all Packet Length PORDs.**
- **Second, we consider all Uni-Burst Size PORDs.**
- **Third, Uni-Burst Time PORDs are inserted.**
- **Then, we put all Uni-Burst Count PORDs.**
- **Next, we take all Bi-Burst Size PORDs.**
- **Finally, Bi-Burst Time PORDs are considered.**

We insert the above PORDs in the order of their appearances in the trace.

D. Example

We give an example to clarify our approach. Figure 4 depicts a website trace where packet sequences between Tx and Rx are shown. Each packet in the figure has size s in bytes and time t. Time t is the number of seconds since Epoch (1 January, 1970). Times shown in this example are for illustration purposes only. We set the time for the first packet in the trace to zero to have

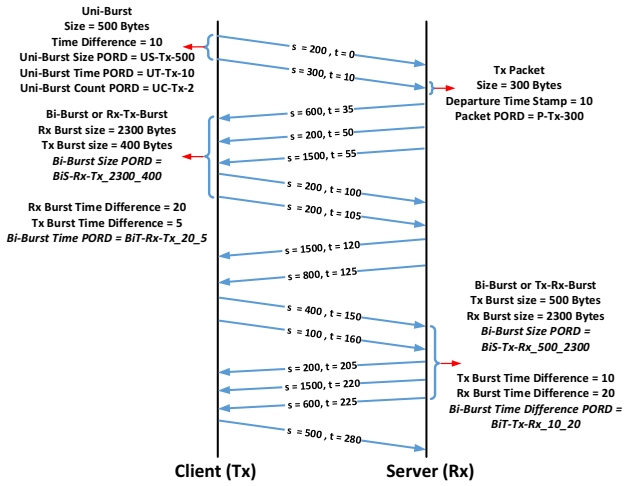


Fig. 4: Example of how P2V generates PORDs (Packet Words) from a trace.

it as a reference. Figure 4 shows a uni-burst example of size 500 (200 plus 300) and time difference of 10 (10 minus 0). Figure 4 illustrates some PORDs constructed by the P2V model.

E. Classification

The input to GloVe is the PORPUS (for Packet cORPUS) which is a collection of POCUMENTS used for training the GloVe model. GloVe produces word (PORD) real-valued vectors. Now we can use these PORD vectors as features in website fingerprinting classification. For each trace (from training set or testing set), we construct a *Trace Vector* by averaging all PORD vectors in that trace. It is worth mentioning that the testing set traces are not used to train the GloVe model to avoid overfitting. Notice that the trace vector is a fixed-length (d-dimensional) vector as every PORD is originally a d-dimensional vector. These trace vectors are used for the regular machine learning classification task where we classify using the naïve-bayes (NB) algorithm. In our evaluation, we show how the trace vectors improve the website fingerprinting attack.

IV. Evaluation

In this section, we present our evaluation of the proposed approach to perform the website fingerprinting attack. We first discuss the dataset used and then the experimental results.

A. Dataset

We use a dataset collected by Liberatore and Levine [4]. We call it the *HTTPS* dataset. The traces were collected while browsing websites using HTTPS protocol. As this dataset has been widely used in previous studies, this enables us to compare the results of our approach with other techniques. The traces have been collected for over two months using 2000 websites.

B. Experimental Results

We now present the results of our experiments. For each experiment, we varied the number of selected websites between 20, 40, 60, 80, and 100. To train GloVe, we use 64 randomly selected traces from each website to build the model. We use the P2V approach described in Section III to produce PORD vectors. The GloVe word vector size we use is 300. We experiment with a context window of 8. For the gradient descent algorithm, GloVe trains the model using AdaGrad [22]. We use an initial learning rate of 0.05.

For the machine learning classification task, we use 16 randomly selected traces per website (class) for training the classifier, and 4 randomly selected traces per class for testing. We generate trace vectors as described in Section III-E. To avoid overfitting, none of the testing set traces is used to build the GloVe model. Each experiment has been run ten times with the websites randomly selected from the pool of websites in each run. Then the average accuracy has been obtained.

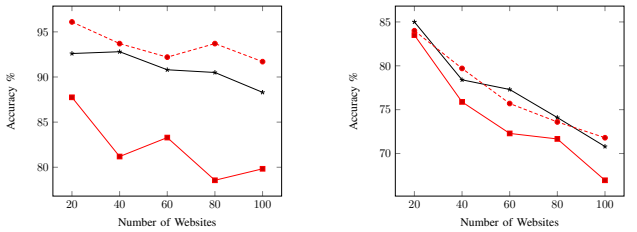
In order to evaluate the performance of our approach, we considered three of the most effective defense mechanisms as well as no applied defense. These defenses are (1) Pad To MTU. (2) Direct Target Sampling. (3) Traffic Morphing. The reason we choose these defenses over other packet delay defenses was justified in Section II-C.

We compare our results with state-of-the-art website fingerprinting classifiers. These classifiers are VNG++ [17] and Panchenko [18]. There are other classifiers in the website fingerprinting literature such as LL [4], OSAD [19] and others. However, recent studies concluded that VNG++ and Panchenko are two of the most accurate classifiers. As indicated in [17], VNG++ performs better than LL. Also, a recent study [23] shows that Panchenko outperforms the OSAD classifier. VNG++ classifier uses total website upload time, uplink and downlink bandwidth, and uni-bursts as features. It applies the naïve-bayes (NB) classifier to get the prediction. Panchenko classifier uses a large collection of features like packet order, HTML markers, uni-bursts, and others. Panchenko utilizes the support vector machine (SVM) classifier. Our P2V approach produces fixed-length trace vectors that are used in classification. We apply our approach against the two classification methods (VNG++ and Panchenko) using naïve-bayes. We use the *Weka* [24] implementation of these classifiers.

We now present the evaluation by running the experiments against the HTTPS dataset. Figure 5a shows the average accuracy when evaluating HTTPS with no defense considered. The X-axis represents the number of websites where we evaluate the experiments against 20, 40, 60, 80, and 100 websites. The Y-axis represents the ten-experiment average accuracy for each classifier including our P2V one. For example, with 20 websites, the accuracies for VNG++, Panchenko, and P2V are 87.75, 92.6, and 96.1 respectively. Our approach proves to perform well even with large number of classes where it achieves accuracies above 90 %. VNG++ does not perform well even when there is no defense applied.

On the other hand, applying defense techniques to the transmitted packets changes the characteristics of the website traffic distribution. This makes classification harder and more sophisticated methods should be used to extract the right patterns.

This concept is clearly illustrated in Figure 5b. We can see the overall accuracies drop for all classifiers, including P2V, in this figure as compared to Figure 5a where there is no defense applied. As discussed in Section II-C, Pad to MTU defense pads each packet to the maximum size (MTU) which is 1500 bytes. This defense is less used in practice as it incurs more overhead. When every packet is padded to 1500 bytes, the vocabulary for the P2V model is not rich and hence the statistics will not build an accurate

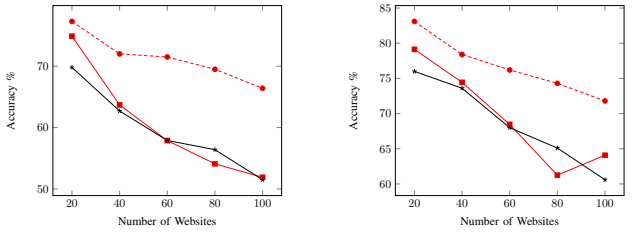


(a) No Defense (b) Pad To MTU

Fig. 5: No Defense and Pad To MTU - HTTPS data: VNG++, Panchenko, P2V.

model. More training data and possibly a more sophisticated model may be required to handle this padding case.

To show our approach resists advanced distribution-based defenses, we run the experiments with Direct Target Sampling and Traffic Morphing. Figures 6a and 6b show the HTTPS dataset results when considering these distribution-based defenses. The figures show the superior performance of the P2V model over the other methods. In DTS, for example, when running the experiments with 60 randomly selected websites, the accuracy for P2V is 71.5 % while for VNG++ and Panchenko, the accuracies are 57.88 and 57.9% respectively.



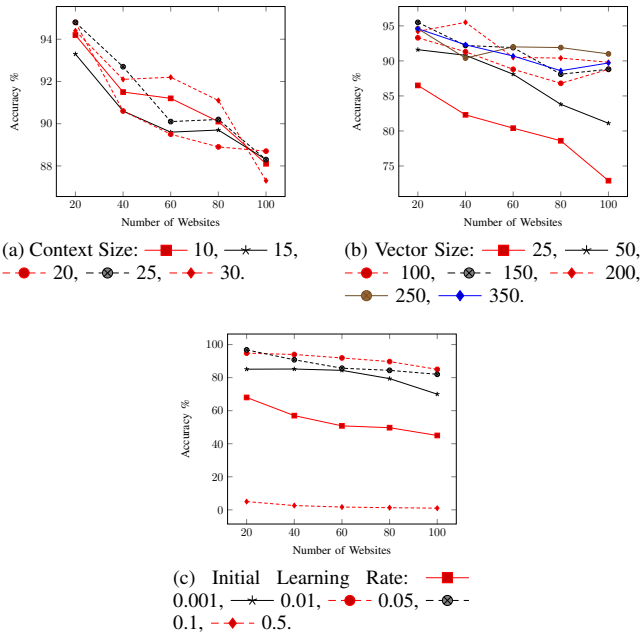
(a) Direct Target Sampling (b) Traffic Morphing

Fig. 6: Direct Target Sampling and Traffic Morphing - HTTPS data: VNG++, Panchenko, P2V.

As discussed throughout the paper, the fact that packet flows in the TCP protocol affect subsequent packet flows helps construct meaningful statistics in the co-occurrence matrix which is the primary source of information to the GloVe model. P2V produces trace vectors that capture the characteristics of the website even though the defender tries to disguise the network packets actual distribution. We notice that Panchenko classifier preforms worse than previous experiments in Figures 5a and 5b. This may be due to the fact that some features used in Panchenko classifier such as HTML Markers do not capture the actual characteristics of morphed packets. When comparing DTS and TM in Figures 6a and 6b, we can see that DTS defense is better than TM as it fools the attacker's classifiers and causes the accuracy to be less. There is a trade-off though. DTS incurs more overhead as it generates more bytes to be padded. In contrast, TM uses convex optimization to lower this cost.

C. Model Analysis

Figure 7 shows the effect of varying context size, vector length, and initial learning rate. We run these experiments when no defense is applied. Varying the context size does not improve the results significantly as compared to the other two parameters, vector length and initial learning rate. Small vector lengths result in low-dimensional trace vectors that do not help the machine learning classifier much. On the contrary, large initial learning rates give extremely bad results. This is because the gradient descent algorithm does not learn the word vectors well which will make the P2V model produce bad trace vectors.



(a) Context Size: 10, 15, 20, 25, 30.

(b) Vector Size: 25, 50, 100, 150, 200, 250, 350.

(c) Initial Learning Rate: 0.001, 0.01, 0.05, 0.1, 0.5.

Fig. 7: P2V Model Analysis - HTTPS data - No defense.

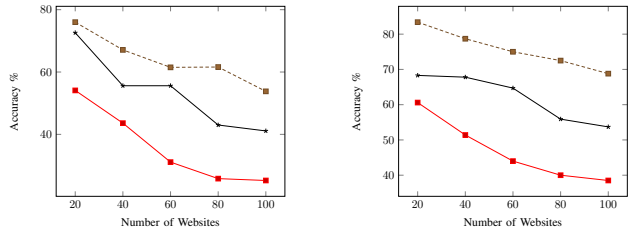
V. Discussion

Website fingerprinting is the ability for the attackers to identify the websites accessed by users. Attackers may be tyrannical governments who try to suppress freedom. However, attackers may be organizations or even governments who try to track malicious activities. Defenders such as Tor anonymity network apply countermeasures (or defenses) to hinder the attackers ability to threaten web navigation privacy.

Previous studies on website fingerprinting used features such as packet size with direction of data transmission, the length of combined sequential packets in the same direction, called burst, and time taken to load the webpage. These features are extracted from a trace of network traffic belonging to a single website. Moreover, previous studies assumed independence between features extracted. In this paper, we model the website fingerprinting attack using GloVe (a word vector representation model). Communication between ends is stacked over the TCP protocol. TCP uses flow control mechanism to ensure certain understanding between client and server. We view this understanding as a dialogue between two ends. In TCP, each packet flow affects the subsequent packet flow. This means there is a dependence between consecutive packet flows. We use this fact to build a packet-packet co-occurrence matrix which is the main source of information used by the GloVe model.

Figure 8 shows the effect of fewer vocabulary (PORDs) with DTS and TM defenses. (1) uses Packet Length, Uni-Burst Size,

and Uni-Burst Time PORDs only. (2) uses Packet Length, Uni-Burst Size, Uni-Burst Time, Uni-Burst Count, and Bi-Burst Size PORDs. It is clear from the results how enriching the vocabulary improves the P2V classifier. (3) uses the same PORDs as in (2) but with the ACK packets included. We can see how the ACK packets improve the P2V model. These packets can be viewed as filter or stop words in NLP where they are considered crucial for some tasks like Author Attribution as they retain signatures of author style [13]. Furthermore, GloVe uses the context-counting approach where the stop words are considered with a weighting function that limits the effect of frequent co-occurrences. This is the first time study which considers the ACK packets in the website fingerprinting attack design.



(a) DTS: 1, 2, 3. (b) TM: 1, 2, 3.

Fig. 8: DTS and TM when increasing the vocabulary and considering the ACK packets - HTTPS data.

VI. Conclusion

To advance the defense in website fingerprinting, we propose the P2V model which views network packets as words. We showed how the proposed model can be used to improve the website fingerprinting accuracy and defeat the existing defenses. Our experimental results show that our proposed attack can achieve higher accuracy in identifying a website from a given trace, than claimed in previous studies. The experimental results show also our new approach is more resilient to website fingerprinting defenses than previous works.

VII. Acknowledgment

This material is based upon work supported by National Science Foundation under Award No. CNS 1229652 and The Air Force Office of Scientific Research under Award No. FA9550-14-1-0173.

References

[1] Elena Zheleva and Lise Getoor. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th international conference on World wide web*, pages 531–540. ACM, 2009.

[2] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[3] Carlton R Davis. *IPSec: Securing VPNs*. McGraw-Hill Professional, 2001.

[4] Marc Liberatore and Brian Neil Levine. Inferring the source of encrypted http connections. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 255–263. ACM, 2006.

[5] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 31–42. ACM, 2009.

[6] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. Touching from a distance: Website fingerprinting attacks and defenses. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 605–616. ACM, 2012.

[7] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective attacks and provable defenses for website fingerprinting. In *Proc. 23th USENIX Security Symposium (USENIX)*, 2014.

[8] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. A systematic approach to developing and evaluating website fingerprinting defenses. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 227–238. ACM, 2014.

[9] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[11] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[12] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *ICLR Workshop*, 2013.

[13] Arun, Saradha, V. Suresh, Murty, and C. E. Veni Madhavan. Stopwords and Stylometry : A Latent Dirichlet Allocation Approach. In *NIPS Workshop on Applications for Topic Models: Text and Beyond*, Whistler, Canada, 2009.

[14] Qixiang Sun, Daniel R Simon, Yi-Min Wang, Wilf Russell, Venkata N Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 19–30. IEEE, 2002.

[15] Andrew Hintz. Fingerprinting websites using traffic analysis. In *Privacy Enhancing Technologies*, pages 171–178. Springer, 2003.

[16] Mashael AlSabah, Kevin Bauer, and Ian Goldberg. Enhancing tor's performance using real-time traffic classification. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 73–84. ACM, 2012.

[17] Kevin P Dyer, Scott E Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 332–346. IEEE, 2012.

[18] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, pages 103–114. ACM, 2011.

[19] Tao Wang and Ian Goldberg. Improved website fingerprinting on tor. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, pages 201–212. ACM, 2013.

[20] Charles V. Wright, Scott E. Coull, and Fabian Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *In Proceedings of the 16th Network and Distributed Security Symposium*, pages 237–250. IEEE, 2009.

[21] V. Paxson and M. Allman. Computing tcp's retransmission timer, 2000.

[22] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011. ISSN 1532-4435.

[23] Brad Miller, Ling Huang, Anthony D Joseph, and J Doug Tygar. I know why you went to the clinic: Risks and realization of https traffic analysis. In *Privacy Enhancing Technologies*, pages 143–163. Springer, 2014.

[24] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1): 10–18, 2009.