Hindawi
Security and Communication Networks
Volume 2021, Article ID 3265300, 17 pages
https://doi.org/10.1155/2021/3265300

Research Article

PF: Website Fingerprinting Attack Using Probabilistic Topic Model

Hongcheng Zou[1] **Ziling Wei**[1] **Jinshu Su**[1] **Baokang Zhao**[1] **Yusheng Xia**[1] **and Na Zhao**[1,2]

[1]*College of Computer, National University of Defense Technology, Changsha, Hunan, China*
[2]*Department of Information Science and Technology, Changsha Normal University, Changsha, Hunan, China*

Correspondence should be addressed to Ziling Wei; weiziling@nudt.edu.cn

Received 11 May 2021; Accepted 27 September 2021; Published 19 October 2021

Academic Editor: Fazlullah Khan

Website fingerprinting (WFP) attack enables identifying the websites a user is browsing even under the protection of privacy-enhancing technologies (PETs). Previous studies demonstrate that most machine-learning attacks need multiple types of features as input, thus inducing tremendous feature engineering work. However, we show the other alternative. That is, we present Probabilistic Fingerprinting (PF), a new website fingerprinting attack that merely leverages one type of features. They are produced by using a mathematical model PWFP that combines a probabilistic topic model with WFP for the first time, due to a finding that a plain text and the sequence file generated from a traffic instance are essentially the same. Experimental results show that the proposed new features are more distinguishing than the existing features. In a closed-world setting, PF attains a better accuracy performance (99.79% at most) than prior attacks on various datasets gathered in the scenarios of Shadowsocks, SSH, and TLS, respectively. Besides, even when the number of training instances drops to as few as 4, PF still reaches an accuracy of above 90%. In the more realistic open-world setting, PF attains a high true positive rate (TPR) and Bayes detection rate (BDR), and a low false positive rate (FPR) in all evaluations, which outperforms the other attacks. These results highlight that it is meaningful and possible to explore new features to improve the accuracy of WFP attacks.

1. Introduction

Nowadays, privacy is one of the most important concerns for online users. Hence, privacy-enhancing technologies (PETs) like Shadowsocks [1], SSH, etc., have been leveraged to guarantee people's privacy, including those criminals who engage in illegal online activities. These unlawful activities severely impair society. For instance, in just two and a half years, the scale of illicit transactions in the black market site "Silk Road" reaches about 1.5 billion U.S. dollars, gathering more than 4,000 illegal merchants and 150,000 anonymous users. A literature survey reveals that a website fingerprinting (WFP) attack can detect these activities by inferring the websites being visited. Hence, WFP plays an important role in fostering a peaceful society that is free of fear and violence. This goal is a part of the 17 sustainable development goals (SDGs) accepted by the United Nations General Assembly in 2015 [2].

The primary idea of WFP attacks can be summarized as follows. A local eavesdropper (i.e., an attacker) listens on the wire and intercepts the target user's network traffic. After that, he trains a classifier according to the statistical features of the traffic. Note that the features generally contain packet length, timing information, order information, and so on. Finally, the attacker could leverage the classifier to identify the surfing websites of the user. The possible target user under WFP attacks might be anyone who is surfing the Internet even under the protection of PETs.

To monitor and stop online criminal activities, as of today, researchers have proposed various attacks by utilizing traditional machine-learning and deep-learning methods to undermine all kinds of PETs. Specifically, the former methods cover naive Bayers (NB) [3, 4]; support vector machine (SVM) [5, 6]; and edit distance [7, 8], random forest (RF) [9, 10], K-nearest neighbor (KNN) [11, 12], hidden Markov model (HMM) [13], and so on. The latter methods leverage different deep neural networks (DNN), such as stacked denoised autoencoder (SDAE) [14], convolutional neural networks (CNN) [15, 16], and long short-term memory (LSTM) [17], to automated extract features. Generally speaking, most traditional machine-learning attacks need to leverage multiple types of features to reach an expected accuracy. For example, the KNN attack proposed by Wang et al. uses six types of features [18].

As known to all, more types of features result in more tedious feature engineering work for WFP, which is unlikable. To avoid such annoying jobs, researchers have turned to deep-learning techniques for help. Previous studies show that deep-learning attacks (e.g., Abe_SDAE [14], DF (deep fingerprinting) [19], and Tik-Tok [20]) usually utilize one type of feature, such as packet direction, and achieve a satisfying accuracy performance, which is better than that of traditional machine-learning attacks [19, 20]. The reason for the difference between the two kinds of attacks is probably ascribed to their different ability of automatic feature learning. Thus, we have the comprehension that introducing more types of features is not indispensable for reaching a good performance.

Unfortunately, although deep-learning attacks can avoid the tedious feature engineering work, they generally need a lot of computing resources, which require an additional budget. Besides, previous research also indicates that a considerable number of training samples are necessary for deep-learning attacks to obtain an expected accuracy [16]. Inevitably, gathering enough training samples will consume a lot of time. It is even a much more unpleasant and hard work. On top of that, considering that a WFP attack should frequently retrain its model to face the challenge of data staleness problem, the work of data gathering becomes heavier and tougher for a deep-learning attack.

In this case, traditional machine-learning attacks become meaningful and essential for WFP. Hence, the second alternative to reduce the work of feature engineering is to find out one type of more effective features, which is the aim of representational learning. Thus, it is interesting to investigate whether it is possible to reach a well-pleasing accuracy for a traditional machine-learning attack only using one type of features.

To the best of our knowledge, there already exist two traditional machine-learning attacks (i.e., CUMUL [5], PHMM [13]) that only take one type of features as input. Unfortunately, they are inferior to deep-learning attacks in accuracy performance [19]. By careful dimensional analysis, we note that creating new features in PHMM and CUMUL does not involve dimensional change. In other words, the new features have the same physical significance (i.e., a length) as the existing features for the two attacks. This truth possibly explains why CUMUL and PHMM perform worse than deep-learning attacks. In this case, it is meaningful for us to devise a type of features with some different physical significance.

Thus, in this work, we propose a new type of feature, i.e., topic probability vector, which is demonstrated to be highly effective. The proposed features have a different physical significance (i.e., a probability) from the existing features (i.e., a length). The new type of features are obtained by the PWFP model, which combines the typical probabilistic topic model, namely, Probabilistic Latent Semantic Index (PLSI), with WFP. Based on the new features, the Probabilistic Fingerprinting (PF) attack is proposed and evaluated. Evaluation results show that PF performs better than a deep-learning attack (i.e., DF) while using fewer features. This work is the first to indicate that a traditional machine-learning attack can beat a deep-learning attack.

The major contributions and novelties of this paper are summarized as follows:

(1) For the first time, we reveal the similarity of a plain text and the sequence file of a traffic instance in essence. Inspired by the finding, we create one type of features, i.e., topic probability vector, each component of which has a special physical significance, namely, a probability. The new features are obtained by the PWFP model, which is based on PLSI. To the best of our knowledge, it is the first time to leverage the probabilistic topic model for WFP.

(2) We propose PF, which first introduces PLSI for WFP and creates a topic probability vector for each traffic instance. Based on the obtained vectors, a KNN classifier is applied to perform a website fingerprinting attack. To date, the topic probability vector has never been presented and used before. PF has a powerful ability to distinguish traffic instances gathered in various scenarios. It can dramatically reduce feature engineering work and the number of features needed while obtaining a better accuracy than a deep-learning attack, namely, DF. As far as we know, it is the first time that a traditional machine-learning attack beats a deep-learning attack.

(3) We show the superiority of the proposed type of features over the existing features by comparison evaluation. The effectiveness of different attacks is evaluated in the closed-world evaluations against various traffic, including Shadowsocks, SSH, and TLS. Amongst all, PF performs the best. We also experiment on how the number of training instances affects the accuracy. Results show that PF only needs as few as four training instances to reach an accuracy of over 90%, which beats others. This advantage is useful for addressing the data staleness issues in WFP.

(4) In the open-world evaluation, we use the Precision-Recall curves to compare different attacks' performances to avoid the base-rate fallacy. PF all achieves a high recall and precision, which substantially overwhelms the other attacks. Besides, we investigate the impact of different ratios of the number of unmonitored training instances to the number of total unmonitored instances on TPR, FPR, and BDR. PF works best in all situations. Our experiments also indicate the excellent performance of PF against defended datasets.

Organization. The remainder of this paper is organized as follows. In Section 2, we survey prior research work.

Subsequently, Section 3 describes the threat model of this work. Furthermore, the key techniques of the PF attack are explained in Section 4. To test our attack, Section 5 presents the experimental preparation. Then, in Section 6, we evaluate the PF attack in different scenarios and present the results, respectively. Finally, we make a deep discussion in Section 7 and conclude the whole paper in Section 8.

2. Related Work

This section first surveys different kinds of significant WFP attacks, including resource length attacks, traditional machine-learning attacks, and deep-learning attacks. Then, we categorize and summarize prior work on the main WFP defense methods. Moreover, four representative attacks and two typical defenses selected in the following experimental evaluations are introduced in detail.

2.1. WFP Attacks. The WFP attacks originate from resource length attacks, which utilize the length of web page resources to identify a web page. In HTTP1.0, web page resources (images, scripts, etc.) are each requested with a separate TCP connection. Thus, the total length of each resource can be identified by distinguishing different connections. The earliest prototype of the resource length attack was designed and implemented by Cheng and Avnur [21]. Similar research followed later [22–24]. With the emergence of HTTP1.1 and various PETs, the performance of resource length attacks decreases sharply. Hence, researchers dig out more and more new features from traffic to improve the accuracy performance.

With the help of traditional machine-learning and deep-learning techniques, the success rate of WFP attacks rises greatly. If an attacker uses a traditional machine-learning method to make predictions on the websites, the attack can be classified into a traditional machine-learning attack. Such typical attacks include Li-NB [4], Li-Jaccard [4], OSAD [7], DLSVM [8], Pa-SVM [6], He-SVM [25], CUMUL [5], KNN [18], WPF [11], KFP (K-fingerprinting) [26], and PHMM (profile hidden Markov model) [13]. These attacks leverage different traditional machine-learning techniques, such as Bayers classifier, Jaccard coefficient, SVM, KNN, RF, and HMM. The major disadvantage of traditional learning attacks lies in their requiring heavy work of feature engineering. To avoid this shortcoming, researchers need to manually create a type of features that are highly effective.

Considering the excellent performance of deep-learning methods, people have introduced them into the WFP area lately. In deep-learning attacks, the training dataset is absorbed to learn the parameters of deep neural networks, which can then be used to classify the test dataset. The deep-learning attacks have thrived since Abe and Goto first studied the application of stacked denoising autoencoders (SDAE) in WFP attacks [14]. In recent years, different neural networks, such as SDAE, LSTM, CNN, were leveraged by various deep-learning attacks, including DF [19], var-CNN [15], Tik-Tok [20], and AWF [17]. Although deep-learning attacks reach a high accuracy performance, they have a high demand for training data scale. Also, the attacker needs a substantial budget, which significantly limits the application of deep learning attacks.

To better evaluate our attack, we have selected four typical attacks for comparison, namely, KNN [18], KFP [26], DF [19], and PHMM [13]. They use different techniques and are commonly selected as benchmarks in the WFP literature. These attacks are briefly introduced as follows.

2.1.1. KNN. The KNN classifier was presented by Wang et al. with weight adjustment based on a large set of features, as many as 3736 [18]. The weights are used to tune the contributions to the KNN distance of different features. As weight learning proceeds, the KNN distance comes to focus on weights for features that are useful for classification. Due to its good performance in efficiency and accuracy, the attack is extensively used as a benchmark in the WFP area.

2.1.2. KFP. Hayes et al. proposed a KFP attack method based on RF and KNN [26]. The method uses the random forest to extract the fingerprint for each traffic instance, instead of directly using the classification output of the forest. In the open-world setting, they feed these fingerprints to a KNN classifier. Specifically, by computing the Hamming distance of fingerprints, KFP classifies a test instance as the label of the closest k training instances if and only if the k labels are in complete agreement. Otherwise, the test instance would be classified into the unmonitored class.

2.1.3. DF. Sirinam et al. first presented the DF attack [19]. This attack leverages CNN with sophisticated architectural design. To train DF, the authors applied Dropout and Batch Normalization (BN) to prevent overfitting. Also, DF attains a high success rate in both closed-world and open-world settings. It is a typical deep-learning attack.

2.1.4. PHMM. Zhuo et al. proposed the PHMM attack firstly by introducing bioinformatics into the WFP attack [13]. This attack collects the features of packet length with direction from each traffic instance and transforms the feature value of each packet into the alphabet to be recognized by the model. The transformation is named symbolization. In the scenarios of SSH and Shadowsocks, PHMM achieves a good performance.

2.2. WFP Defenses. To defend against WFP attacks, many countermeasures were taken to obfuscate the traffic features by modifying the traffic. The WFP defenses can be classified into packet padding defenses, decoy page defenses, and so on [27, 28]. We further classify the packet padding defenses into two types, namely, packet padding defenses with and without delay. The latency of packet padding defenses depends on their padding strategy. Some defense methods, including maximum padding defense [29], AP (adaptive padding)-based defenses [30], probabilistic defenses [13], and so on, generally introduce very low latency, which can be omitted. The major packet padding defenses with delay include BuFLO (buffered fixed-length obfuscator) [29], CS-BuFLO (congestion sensitive BuFLO) [31], and Tamaraw [32]. Decoy defenses contain two types. One is to mimic a decoy page [33]. The other one is to add a decoy page as the background traffic [6].

Besides the aforementioned defenses, there also exist some other defenses that work at the application layer, such as randomized pipelining, which is embedded in browsers and HTTPOS. The HTTPOS defense was firstly presented by Luo et al. [27]. It needs to modify the HTTP headers and changes the HTTP requests to control the size of packets, which makes the implementation of HTTPOS a little complicated. In addition, Wang et al. presented another defense called Walkie-Talkie [28]. Walkie-Talkie works in the half-duplex mode and needs to add dummy packets and delays to create collisions. It requires both latency overhead and bandwidth overhead.

To evaluate our attack, this study selects two latest probabilistic defenses, namely, probabilistic dummy packet defense and probabilistic MTU (maximum transmission unit) padding defense, to produce the defended datasets. The former enables each packet to insert a dummy packet ahead of it with a given probability, while the latter lets each packet to decide whether to pad its length to MTU or not with a predefined probability. Each packet has the same probability to make its decision in the two defenses. The two probabilistic defenses were first simulated by Zhuo et al. [13]. They both use the probability to weigh between latency and efficacy.

3. Threat Model

Our work mainly focuses on website fingerprinting under the protection of Shadowsocks, SSH, and TLS. These PETs apply different techniques and are commonly used all over the world. To be specific, Shadowsocks is a free and open-source encryption protocol project that is widely used. It is not a proxy on its own but a protocol. Shadowsocks has become increasingly popular according to Google trend in recent years. According to incomplete statistics, hundreds of thousands of people have downloaded the Shadowsocks client [9]. The SSH protocol is included and supported in all operating systems for the reason that telnet and rlogin are insecure. Thus, it is convenient for those people who seek to protect their privacy. Also, the TLS technique is becoming more and more universal. According to the statistical data of Google in February 2021, about 95% of web traffic in Chrome for Mac is encrypted, while 90% of web traffic in Chrome for Windows is encrypted.

Figure 1 shows the typical attack scenario in the WFP area [13, 19, 26]. We also use this scenario in our work. A user browses the websites under the protection of Shadowsocks, SSH, or TLS. A passive local attacker intercepts the encrypted traffic between the user and the communication network entrance and tries to infer the user's browsing privacy. Specifically, the word "passive" means that the adversary can record network packets but not modify, delay, drop, or decrypt them. Besides, the word "local" means that the adversary has access only to the link between the user and the entry of the communication networks. It is noted that all the Shadowsocks, SSH, and TLS traffic is encrypted in a different way. Like previous literature [13, 19, 26], we assume that the adversary has some prior knowledge of the user and only aims at identifying the websites. He does not try to decrypt packets or modify transmissions. Hence, our attack has nothing to do with the encrypted methods of the traffic.

In this work, we study the fingerprinting of the home page of those websites. That is, all instances are obtained from the homepages of websites. This task is called website fingerprinting by most authors in this field. As previous literature has mentioned, the adversary is supposed to be able to isolate and parse each traffic generated by a web page visit. Such isolating and parsing would be done before performing website fingerprinting attacks.

As with prior work, we study two scenarios, namely, closed-world and open-world. To be specific, in a closed-world scenario, it is assumed that the user only visits a given set of websites, namely, monitored websites, whereas in an open-world scenario, the user is allowed to visit not just the monitored websites but a large number of unmonitored websites, namely, the open-world. Apparently, the open-world scenario is of practical interest.

4. THE Proposed PF Attack

This section explains in detail the scheme of PF. For a better understanding, we first give an overview of PF, which presents the data processing flow and the module diagrams in the whole process of identification. In the following subsections, each module is elaborated by further decomposition if needed. To validate the PF scheme, the last subsection introduces the implementation of PF by pseudocode and an example.

4.1. PF Overview. In the scenario of PETs like Shadowsocks, SSH, and TLS, we create a new type of features based on an existing type of features, i.e., packet length with direction. Based on the new features, we put forward a new attack PF, whose basic principle is shown in Figure 2. At the very beginning of the PF, the attacker needs to gather datasets in different scenarios of PETs. Then, the datasets are put into the framework of PF, which contains three basic modules, including preprocessing datasets, proposing new features, and classifying.

Specifically, the first module needs to perform the symbolization to produce the sequence files for all the instances and do the TF-IDF transformation to produce the representative vector of each instance. The vectors are taken as the input of the PWFP model. The second module leverages two submodules, including model training and fold-in process, to obtain the proposed new features of training instances and test instances, respectively. Finally, we use the new features to perform classification in the third module. Note that KNN is used as the classifier in this work. The technical details of these modules are explained in the following subsections.

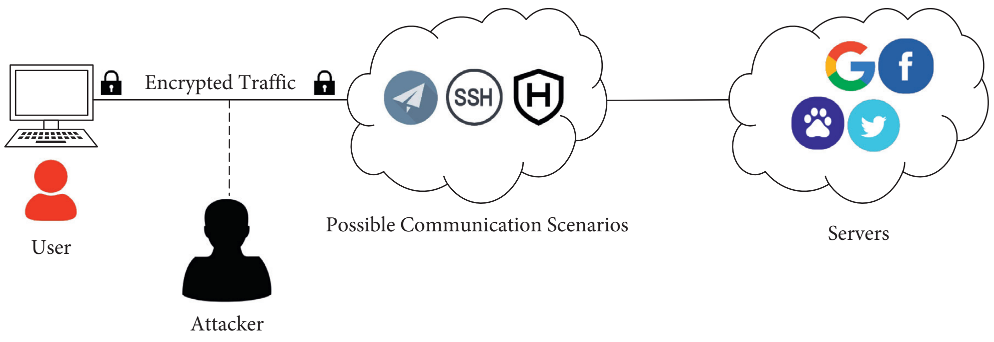


Figure 1: The typical threat model in different communication scenarios, e.g., Shadowsocks, SSH, and HTTPS.

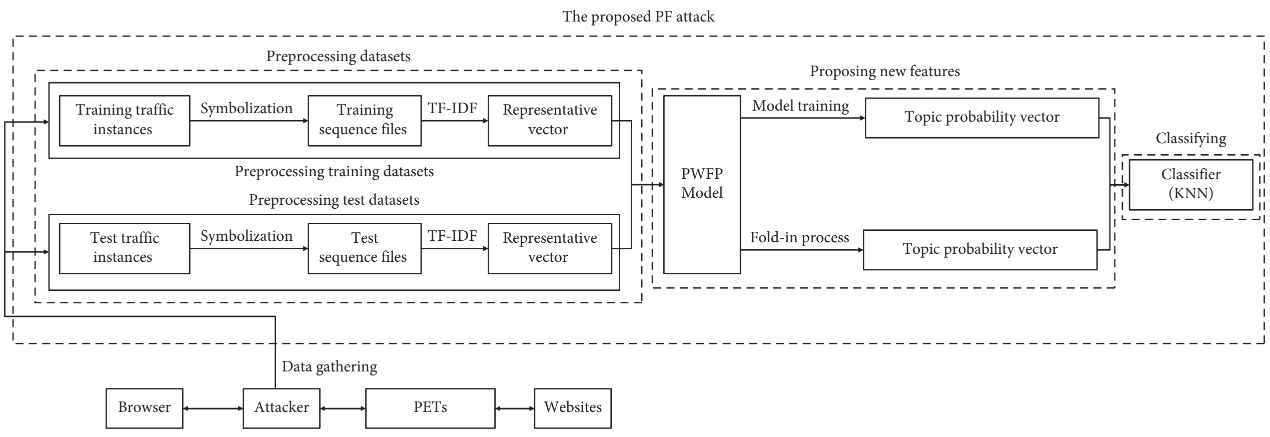


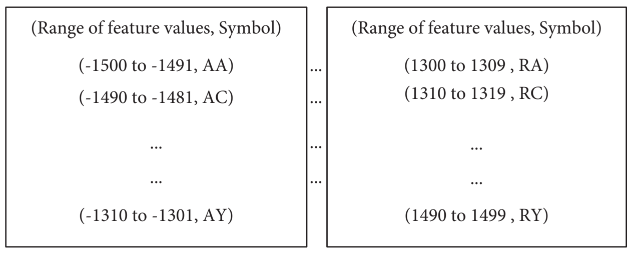
Figure 2: The basic principle of the PF attack.

4.2. Preprocessing Datasets. In this subsection, we preprocess the training and test instances to fit the PWFP model. Specifically, the TF-IDF (Term Frequency-Inverse Document Frequency) transformation, which has been extensively used in the field of text classification, is then applied to obtain a good data representation for the traffic instances.

To do the TF-IDF transformation, we need to construct a connection between a traffic instance and a plain text. We build the connection by the following steps. Firstly, we introduce the concept of symbolization by which each traffic instance is converted into a traffic sequence file. Then, we reveal the similarity between a plain text and the sequence file of a traffic instance for the first time. Based on the finding, we can view each traffic instance as a plain text naturally.

4.2.1. Symbolization. At the very beginning of preprocessing, each traffic instance, i.e., a series of consecutive packets generated from a complete web page visit, should be converted into a sequence file. The conversion is named symbolization, as shown in Figure 3.

At first, each packet size with direction should be converted into a feature value (e.g., −1500). To be specific, the packet size decides the quantity of the feature value. Besides, the packet direction determines whether or not the feature value is positive. Since the packet size is no more than 1500 bytes, the feature value of each packet can be defined by



Note: (Range of feature values, Symbol) indicates that the feature values in the given range should be converted into the given symbol after doing symbolization. Each symbol is composed by two letters in HMMER Alphabet, namely {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}.

Figure 3: An intuitive illustration of symbolization.

a number ranging from −1500 to 1500. After doing symbolization, the feature value of each packet in a traffic instance (e.g., −1500) can be further turned into the corresponding symbol (e.g., AA), as illustrated in Figure 3. As we see from the figure, the feature values within a certain range are denoted by two given letters [13]. It is noted that the letters used for symbolization stem from 20 built-in amino acids in the HMMER tool [34]. Thus, the number of optional letters equals 20.

4.2.2. Our Finding. After doing the symbolization, the sequence files are obtained. To apply the probabilistic topic model, we reveal the similarity between a plain text and the sequence file of a traffic instance by comparing two specific examples, as shown in Figure 4. That is, we randomly select a plain CNN news text and a sequence file in our experiments as examples. To the best of our knowledge, it is the first time that the latent similarity is uncovered.

According to the symbolization method mentioned above, the feature value of each packet would be turned into a symbol, which is notated as a "word" in this work. Hence, each sequence file comprises of a lot of "words." On the other side, a plain text comprises many meaningful symbols inside, called words. The similarity in essence between a sequence file and a plain text can be concluded by the following comparison analysis.

At the very beginning, we essentially analyze the similarity of a "word" in a sequence file with a word in a text. On the one hand, the essence of both a word in a text and a "word" in a sequence file is a kind of symbol. On the other hand, similar to a single word in a text, each "word" in a sequence file also has its meaning, which indicates the size and direction of the corresponding packet. Therefore, each "word" in a sequence file is analogous to a word in a text. One more step further, a text is a combination of words. Similarly, a sequence file is a combination of "words". Given the above, each sequence file is analogous to a text. The above analysis is intuitively shown in Figure 4.

For the similarity between a sequence file and a plain text, it is natural to leverage text classification methods for website fingerprinting. Thus, the PWFP model, which incorporates the extensively used text classification method PLSI with WFP, is proposed.

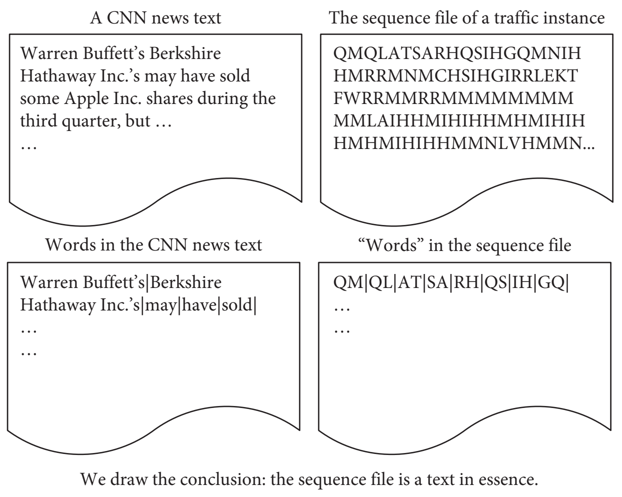

A CNN news text
The sequence file of a traffic instance
Warren Buffett's Berkshire Hathaway Inc.'s may have sold some Apple Inc. shares during the third quarter, but …
…
QMQLATSARHQSIHGQMNIH HMRRMNMCHSIHGIRRLEKT FWRRMMRRMMMMMMMM MMLAIHHMIHIHHMHMIHIH HMHMIHIHHMMNLVHMMN…
Words in the CNN news text
"Words" in the sequence file
Warren Buffett's|Berkshire Hathaway Inc.'s|may|have|sold|
…
…
QM|QL|AT|SA|RH|QS|IH|GQ|
…
…
We draw the conclusion: the sequence file is a text in essence.

Figure 4: The comparison of a plain CNN news text and the sequence file of a specific traffic instance.

4.2.3. TF-IDF Transformation. As mentioned above, we will get the sequence file of each traffic instance after symbolization. However, the sequence files cannot be input into the PWFP model directly. To launch the PWFP model, the sequence file of each traffic instance, including the training instance and test instance, should be represented by a representative vector, namely, the input of the PWFP model. Hence, we also call "a representative vector" as "an input vector." Since the TF-IDF transformation quantifies the importance of each symbol in the traffic sequence files well, we utilize the TF-IDF transformation to produce the input of the PWFP model.

In the process of model training, each training sequence file, e.g., d_i, needs to be converted into the corresponding input vector, namely v_{d_i}, such that it can be fed into the model. The j th component of the input vector is obtained by the TF-IDF transformation of w_j, which is the j th "word" in the sequence file. The TF-IDF transformation of w_j is defined by equation (1).

$$v_{d_i}(j) = tf_{d_i}(w_j) \times idf(w_j), \tag{1}$$

where $tf_{d_i}(w_j)$ and $idf(w_j)$ mean the TF transformation and IDF transformation of w_j, respectively. In detail, $tf_{d_i}(w_j)$ and $idf(w_j)$ are defined as equations (2) and (3), respectively.

$$tf_{d_i}(w_j) = \log\left(1 + \frac{n(d_i, w_j)}{\max\{n(d_i, w_j) \mid j = 1, \ldots, M\}}\right), \tag{2}$$

$$idf(w_j) = \log\left(1 + \frac{N}{m(w_j)}\right), \tag{3}$$

where M denotes the total number of different "words" in the corpus, i.e., all the set of the training sequence files. N denotes the total number of training sequence files. $n(d_i, w_j)$ denotes the number of the "word" w_j occurring in the training sequence file d_i. $m(w_j)$ denotes the number of training sequence files which contain the "word" w_j.

Similarly, the test instances need to perform the TF-IDF transformation before the fold-in process. Each test sequence file, e.g., q_i, needs to be converted into a corresponding input vector, namely, v_{q_i}. The related equations are shown below.

$$\begin{aligned} v_{q_i}(j) &= tf_{q_i}(w_j) \times idf(w_j), \\ tf_{q_i}(w_j) &= \log\left(1 + \frac{n(q_i, w_j)}{\max\{n(q_i, w_j) \mid j = 1, \ldots, M\}}\right), \end{aligned} \tag{4}$$

where $n(q_i, w_j)$ denotes the number of the "word" w_j occurring in the test sequence file q_i.

4.3. Proposing the New Features. After the TF-IDF transformation, each sequence file will generate a representative vector. Since the vector is taken as the input of the following model, it is also called the input vector. The proposed PF attack leverages the PWFP model to process the input vectors. PLSI, which associates a latent semantic variable (i.e., topic) with each observation [35], is the mathematical basis of the PWFP model. After processing the input vectors, the parameters of the PWFP model are solved, which lays a solid foundation for proposing new features. Thus, in this section, the basic theories of PWFP, including the basis of the PWFP model, model training, and fold-in process, will be detailed at first. Lastly, we will propose a type of new features based on the obtained PWFP model.

4.3.1. The Basis of the PWFP Model. In this part, we show the basis of PWFP. Similar to using PLSI for the task of text classification, the PWFP model also introduces a latent variable called "topic," which has the same functions as the latent variable (i.e., topic) in PLSI. Rather than an intelligible meaning in the PLSI model, the variable "topic" in the PWFP model has an abstract meaning. In PWFP, "topic" is an intermediate concept that associates a sequence file (i.e., "text") with a symbol (i.e., "word") therein. The interrelations between a sequence file, a "topic", and a "word" are represented as conditional probabilities. The crux of the PWFP model is to figure out these conditional probabilities, namely, PWFP parameters. To estimate these conditional probabilities, the EM (Expectation-Maximization) algorithm is extensively used.

Once the conditional probabilities are obtained, each traffic instance can be represented by a "topic" probability vector. Each component of the vector indicates the probability of the sequence file (i.e., "text") belonging to the corresponding "topic." Hence, mathematically speaking, the PWFP model can be viewed as a multidimensional space transformation. To better understand the PWFP model, we define some notations as shown in Table 1. Note that we add quotation marks when describing the concepts (e.g., text, topic, word) in WFP to differentiate the same concepts in the field of text classification.

To begin with, we show the graph model of PWFP in Figure 5. Thus, in terms of a generative model, the PWFP model can be built up in the following way:

(1) Select a training sequence file d with probability $p(d)$,
(2) Pick a "topic" z with probability $p(z|d)$,
(3) Generate a "word" w with probability $p(w|z)$.

After the above three steps, we obtain an observed pair (d, w), and the latent variable z is discarded. Hence, for given values of d_i and w_j, the joint probability of $p(d_i, w_j)$ can be deduced by

$$p(d_i, w_j) = p(w_j|d_i)p(d_i) = \sum_{k=1}^{K} p(w_j|z_k)p(d_i|z_k)p(z_k). \tag{5}$$

By repeating the above process, we get all the training sequence files, i.e., the corpus. Hence, the generative probability of the corpus D, namely $L(D, W)$, can be represented as joint probabilities of all the observation pairs. Furthermore, $L(D, W)$ is calculated by

$$L(D, W) = \prod_{i=1}^{N}\prod_{j=1}^{M} p(d_i, w_j)^{n(d_i, w_j)}. \tag{6}$$

After taking the logarithm of $L(D, W)$, $L'(D, W)$ is obtained by

$$L'(D, W) = \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j)\log(p(d_i, w_j)). \tag{7}$$

By combination with equations (5) and (7), $L'(D, W)$ is further deduced to

$$L'(D, W) = \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j)\log\left(\sum_{k=1}^{K} p(w_j|z_k)p(d_i|z_k)p(z_k)\right). \tag{8}$$

The goal of model training is to estimate the probabilities $p(z_k)$, $p(d_i|z_k)$, and $p(w_j|z_k)$, namely, PWFP parameters. To figure out the probabilities, it is necessary to do maximum likelihood estimation for the function $L'(D, W)$. Only when the function $L'(D, W)$ reaches the maximum, the probabilities are optimal. Intuitively, the probabilities can be figured out by letting the derivative of $L'(D, W)$ equal zero. The EM algorithm is introduced to solve the optimization problem.

The EM algorithm includes two steps, i.e., E (expectation) step and M (maximization) step. The two steps take turns to execute until the PWFP parameters converge.

In the E-step, the posterior probability, $p(z_k|d_i, w_j)$, is introduced. It can be computed based on the current estimates of the PWFP parameters, i.e., $p(z_k)$, $p(d_i|z_k)$, and $p(w_j|z_k)$, according to equation (9) derived by the Bayes rule.

$$p(z_k|d_i, w_j) = \frac{p(z_k)p(d_i|z_k)p(w_j|z_k)}{\sum_{l=1}^{K} p(z_l)p(d_i|z_l)p(w_j|z_l)}. \tag{9}$$

In the M-step, the PWFP parameters are updated by the posterior probabilities that are computed in the previous E-step. The equations for undating the parameters can be derived by the Lagrange multiplier method and demonstrated as

$$\begin{cases} p(w_j|z_k) = \dfrac{\sum_{i=1}^{N} n(d_i, w_j)p(z_k|d_i, w_j)}{\sum_{j=1}^{M}\sum_{i=1}^{N} n(d_i, w_j)p(z_k|d_i, w_j)}, \\[2ex] p(d_i|z_k) = \dfrac{\sum_{j=1}^{M} n(d_i, w_j)p(z_k|d_i, w_j)}{\sum_{j=1}^{M}\sum_{i=1}^{N} n(d_i, w_j)p(z_k|d_i, w_j)}, \\[2ex] p(z_k) = \dfrac{1}{R}\sum_{j=1}^{M}\sum_{i=1}^{N} n(d_i, w_j)p(z_k|d_i, w_j), \\[2ex] R \equiv \sum_{j=1}^{M}\sum_{i=1}^{N} n(d_i, w_j). \end{cases} \tag{10}$$

4.3.2. Model Training. The aim of model training is to estimate the PWFP parameters for the training instances. We use the EM algorithm to achieve this goal. The procedures of

Table 1: The notations used in this work.

Notations	Descriptions
d	a training sequence file
q	a test sequence file
z	a "topic"
w	a "word" in a sequence file
W	The vocabulary, i.e., the set of different "words"
D	The corpus, i.e., the set of training sequence files
Z	The set of "topics"
M	The total number of different "words" in the corpus
M'	The total number of "words" in the corpus
N	The total number of training sequence files
N'	The total number of test sequence files
K	The total number of latent "topics"
$n(d,w)$	The number of the "word" w occurring in the training sequence file d
$n(q,w)$	The number of the "word" w occurring in the test sequence file q
p	Probability
$p(\vert)$	Conditional probability
$p(,)$	Joint probability

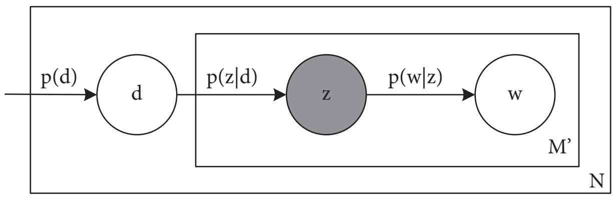


Figure 5: The graph model of PWFP.

using the EM algorithm to estimate the PWFP parameters can be specified as follows:

(1) Initialization: Randomly initialize the posterior probabilities $p(z_k|d_i,w_j)$ and the PWFP parameters of the training instances, namely, $p(d_i|z_k)$, $p(w_j|z_k)$, and $p(z_k)$.

(2) E-step: Based on the current PWFP parameters, update the posterior probabilities $p(z_k|d_i,w_j)$ according to equation (9).

(3)M-step: Based on the current posterior probabilities, update the PWFP parameters of the training instances according to equation (10).

(4) Repeat the E-step and M-step until the number of iterations reaches the setting value. The results in the last iteration are taken as the estimated PWFP parameters for the training instances.

4.3.3. Fold-In Process. Once the PWFP parameters of the training instances are estimated by the EM algorithm, the obtained parameters $p(z_k)$ and $p(w_j|z_k)$ can be further used to infer the parameters of the test instances, i.e., $p(q_i|z_k)$. The inference is called as "fold-in process" in previous work [36]. We also leverage the EM algorithm in the fold-in process. In the fold-in process, the parameters $p(w_j|z_k)$ and $p(z_k)$ remain fixed, whereas the rest parameters $p(q_i|z_k)$ and the posterior probabilities $p(z_k|q_i,w_j)$ need to be updated by the EM algorithm. The fold-in process can be implemented by the following procedures:

(1) Initialization: Randomly initialize the new PWFP parameters, including the conditional probabilities of the test instances $p(q_i|z_k)$ and the posterior probabilities of the test instances $p(z_k|q_i,w_j)$, while keeping the PWFP parameters $p(z_k)$ and $p(w_j|z_k)$ fixed.

(2) E-step: Based on the known PWFP parameters, i.e., $p(z_k)$ and $p(w_j|z_k)$, and the current estimated PWFP parameters, i.e., $p(q_i|z_k)$, update the posterior probabilities $p(z_k|q_i,w_j)$ according to the following equation:

$$p\left(z_k|q_i,w_j\right) = \frac{p(z_k)p(q_i|z_k)p\left(w_j|z_k\right)}{\sum_{l=1}^{K} p(z_l)p(q_i|z_l)p\left(w_j|z_l\right)}. \tag{11}$$

(3) M-step: Based on the current posterior probabilities $p(z_k|q_i,w_j)$, update the parameters $p(q_i|z_k)$ according to the following equation:

$$p(q_i|z_k) = \frac{\sum_{j=1}^{M} n\left(q_i,w_j\right)p\left(z_k|q_i,w_j\right)}{\sum_{j=1}^{M}\sum_{i=1}^{N'} n\left(q_i,w_j\right)p\left(z_k|q_i,w_j\right)}. \tag{12}$$

(4) Repeat the E-step and M-step until the number of iterations reaches the setting value. The results of the last iteration are taken as the estimated PWFP parameters for test traffic instances.

4.3.4. The Proposed New Features. The model training and fold-in process aim to figure out the model parameters, namely, $p(d_i|z_k)$, $p(z_k)$, and $p(q_i|z_k)$. Based on $p(d_i|z_k)$ and $p(z_k)$, each training traffic instance can be represented by a probability vector, each component of which is a "topic" probability. Similarly, each test instance can be represented by a probability vector calculated by $p(q_i|z_k)$ and $p(z_k)$. Hence, in the PF attack, each instance is represented by a vector constituted by "topic" probabilities. Note that the "topic" probabilities are also called the proposed new features.

The proposed new features of a traffic instance and a test instance can be calculated similarly with respective parameters of PWFP. To be specific, the proposed new features of a training instance d_i, namely d_{iv}, can be computed by equation (13). Similarly, the proposed new features of a test instance q_i, i.e., q_{iv}, would be calculated by equation (14):

$$d_{iv} = \left(p(z_1|d_i), \ldots, p(z_k|d_i), \ldots, p(z_K|d_i)\right), \text{where } p(z_k|d_i) = \frac{p(z_k)p(d_i|z_k)}{\sum_{k=1}^{K} p(z_k)p(d_i|z_k)}, \tag{13}$$

$$q_{iv} = \left(p(z_1|q_i), \ldots, p(z_k|q_i), \ldots, p(z_K|q_i)\right), \text{where } p(z_k|q_i) = \frac{p(z_k)p(q_i|z_k)}{\sum_{k=1}^{K} p(z_k)p(q_i|z_k)}. \tag{14}$$

4.4. Classifying. The PF attack leverages typical KNN to make a classification. To achieve this goal, it is needed to select a similarity strategy. The similarity strategy is a way to evaluate the differences between two different feature vectors. The cosine similarity and Euclid similarity are two typical strategies. According to the experimental results, the Euclid similarity is used in our evaluations.

Known from text classification, there usually exist more common words between two texts that belong to the same class than two different classes. Thus, we add a weight coefficient to tune the similarity between two traffic instances or between a traffic instance and a web page class. For two traffic instances, the weight coefficient is defined by the number of common "words" between their respective sequence files. For the similarity between a traffic instance and a web page class, the weight is defined by the average number of the common "words" between the traffic instance and each instance belonging to the web page.

4.5. Implementation. In the above subsections, the theoretical part of PF has been demonstrated extensively. To validate PF, we need to implement all the above techniques step by step, as shown in Algorithm 1. The PF algorithm takes the folder paths of training data and test data as input, and output various metrics in different experiments. The metrics will be discussed in detail in the next section. For convenience, we perform the symbolization before the PF implementation in this work. Note that four parameters are needed for the PF to run, including t, k, m, and n. The parameter t is influenced by the number of web pages, while the parameter k can tune the performance of PF. For the parameters m and n, we set them the same as in our evaluations. According to our experimental results and previous literature [35], the number of iterations can be set as a fixed number 50 in all the evaluations. We also validate the rationality of this setting by a specific experiment.

For a better understanding, we further show the empirical illustration of the PF attack, as shown in Figure 6. Each step is explained with an example of a real dataset in our evaluations.

5. Experimental Preparations

In this section, we make several necessary preparations for the following evaluations. Firstly, we specify the datasets used in our experiments, including the open datasets and our gathered datasets. Then, the metrics of different experimental settings are explained at length. Subsequently, the baseline attacks are briefly introduced.

5.1. Datasets. We use seven datasets in our evaluations, including three open datasets released in Ref. [4, 13], and four datasets collected by ourselves. These datasets are collected in different scenarios, such as Shadowsocks, SSH, and TLS.

5.1.1. The Open Shadowsocks Datasets. We evaluate one open Shadowsocks datasets, i.e., Alexa74 [13], in this work. The Alexa74 dataset contains 74 webpages. Each web page contains 25 instances, 17 of which are for training. The webpages of Alexa74 are filtered from the Alexa top 100 sites.

5.1.2. The Open SSH Datasets. We test two open SSH datasets, namely, SSH55 and SSH100. Both of them are filtered from the Liberatore's dataset [4]. It was collected over an encrypted SSH tunnel for about three months and contains traces of encrypted connections to 2000 sites. Since the dataset has a lot of empty pcap files caused by various failures during the collecting process, we first pick out two datasets with different average lengths of traffic instance file ($15k$ and $20k$) for each web page. Note that all the traffic instance files of each chosen web page are successive in time. Then, we extract the timestamp and length of each TCP packet in each file, and thus obtain the desired datasets.

Besides the open datasets, we gather another four datasets in different scenarios. Three of them are gathered in a Shadowsocks environment, while the rest is gathered in a TLS scenario. The gathering method is specified in the following.

5.1.3. Data Gathering. We rent a cloud server to collect our datasets. To collect the Shadowsocks datasets, the Clash software is installed as the client to communicate with the remote Shadowsocks proxy server, through which we can directly connect to the websites. The datasets are collected automatically by a C crawling script. The script simulates the user's behavior of surfing websites by controlling the Firefox Browser 76.0.1, whose cache function is disabled to prevent loading from the cache, and leverages tcpdump to capture the traffic on the wire. Moreover, the script runs on an

```
input: training data, test data
output: different combinations of performance indicators, including (TP, FN, FP, TN, TPR, FPR, BDR, ACC) and (TP, FN, TPR)
(1)  function PF (training_data path, test_data path)
(2)      set the number of iterations in the training process, i.e., $m$
(3)      set the number of iterations in the fold-in process, i.e., $n$
(4)      set the number of “topics”, i.e., $t$
(5)      set the number of nearest neighbors for KNN, i.e., $k$
(6)      load the training samples, perform the TF-IDF transformation for training instances
(7)      train the PWFP model
(8)      compute the “topic” probability vectors of training samples, i.e., $p(z|d_{\text{training}})$
(9)      load the test samples, perform the TF-IDF transformation for test instances
(10)     fold-in the test samples
(11)     compute the “topic” probability vectors of test samples, i.e., $p(z|d_{\text{test}})$
(12)     calculate the distance between each training sample and test sample
(13)     perform a KNN classification
(14)     statistic the results
(15) end function
```

ALGORITHM 1: The PF implementation in this work

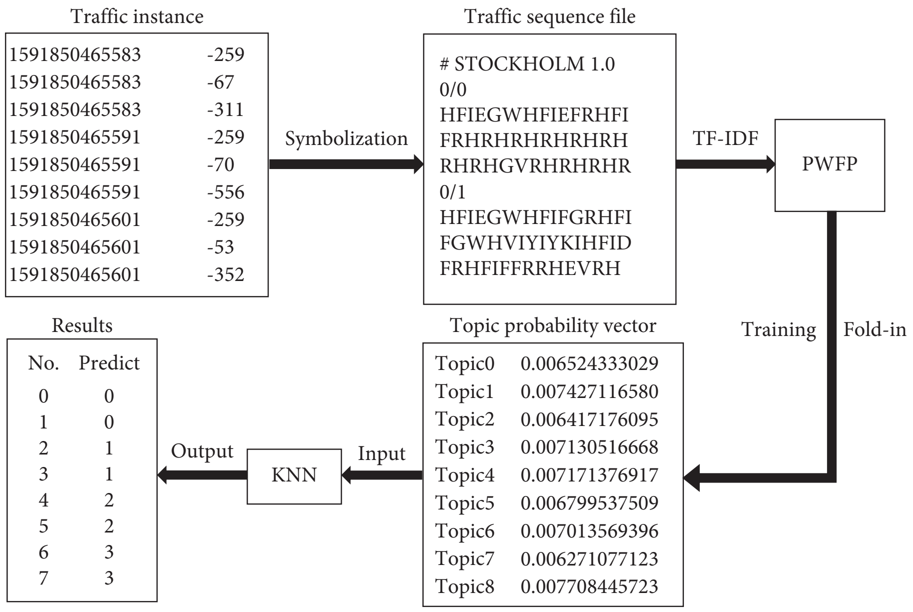


FIGURE 6: The empirical illustration of the PF attack with a specific example.

Ubuntu 16.04 virtual machine to avoid perturbations introduced by the background network traffic. Besides, we disable all the automatic or background network traffic such as the auto-updates. It is also important to make sure that the system-level network settings are all right. For example, it is critical to change the MTU to the standard Ethernet MTU (1500 bytes) and disable offload. We collect three Shadowsocks datasets, i.e., AleSS73, AleSS287, and OpwSS6879. The total size of them is about 23.5 GB, 17.6 GB, and 23.2 GB, respectively.

To collect the HTTPS100 dataset, we follow the same routine mentioned above while closing the Clash software. The total size of HTTPS100 amounts to about 10.4 GB. Note that all the webpages are randomly selected from the Alexa top 10k webpages.

5.1.4. Our Collected Shadowsocks and HTTPS Datasets. The collected datasets, AleSS73, AleSS287, and HTTPS100, contain 73, 287, and 100 webpages, respectively. Each web page has 100, 20, and 40 instances, respectively. They are evaluated in the closed-world setting. Besides, the OpwSS6879 dataset contains 6879 webpages. Each web page has one instance. It is evaluated in the open-world setting together with the AleSS287 dataset. It is worth noting that none of the webpages in AleSS287 is included in the webpages in OpwSS6879.

Table 2: The details of the datasets.

Datasets	PETs	Source	Total Size[1]	Training size	Test size	Evaluations
SSH55	SSH	Open	55×20	55×18	55×2	Closed-world evaluation
SSH100	SSH	Open	100×20	100×18	100×2	Closed-world evaluation
HTTPS100	TLS	Ours	100×40	100×36	100×4	Closed-world evaluation
Alexa74	Shadowsocks	Open	74×25	74×17	74×8	Closed-world evaluation
AleSS73	Shadowsocks	Ours	73×100	73×90	73×10	Closed-world evaluation
AleSS287	Shadowsocks	Ours	287×20	It depends[2]	It depends	Closed-world evaluation
AleSS287 OpwSS6879	Shadowsocks	Ours	287×20 6879×1	It depends	It depends	Open-world evaluation

[1]The total/training/test size column is formatted as web page number × instances per web page. [2]"It depends" means that the training/test size varies according to the specific experiments.

For a better understanding, the datasets used in this work are listed in Table 2.

5.2. Metrics. To evaluate the experimental results, we utilize the following metrics that are extensively used in the WFP area.

In the closed-world evaluation, we use the attacker's accuracy, which is defined as the ratio of the number of correctly classified traces to the total number of traces, to evaluate the performance of different attacks as previous research [13, 19, 26]. The ratio equals TPR, namely, the probability that a monitored web page is classified as the correct monitored web page, in the closed-world scenario. Besides, another indicator called "recall" has the same definition as TPR. In the open-world evaluation, we take into consideration 3 indicators, including TPR, FPR, and BDR:

$$BDR = \frac{TPR \times p(\text{mon})}{TPR \times p(\text{mon}) + FPR \times p(\text{unmon})}, \qquad (15)$$

where $p(\text{mon}) = |\text{monitered test instances}| / |\text{total test instances}|$, $p(\text{un mon}) = 1 - p(\text{mon})$. Note that FPR is defined as the probability that an unmonitored web page is incorrectly classified as a monitored web page. Since BDR considers the differences in the size of the different classes, it is widely used to evaluate the feasibility and effectiveness of an attack [13, 26]. Besides, a metric called "precision" is also used in previous literature [19]. In fact, it can be proved that BDR is equivalent to precision. Thus, we also called BDR as precision in this work.

5.3. State-of-the-Art Attacks. For comparison, we consider four state-of-the-art attacks, including KFP [26], KNN [18], PHMM [13], and DF [19]. Besides, these attacks are based on traditional machine-learning techniques or deep-learning techniques, including RF, ameliorated KNN, HMM, and CNN. Hence, they are representative.

6. Experimental Evaluation

In this section, we first perform two preliminary experiments to adjust the optimal parameters of PF and compare our proposed new type of features with the existing features. To validate the feasibility of PF, the subsequent experiments are conducted under three different scenarios that are extensively studied in previous literature. To be more persuasive, all the Shadowsocks, SSH, and TLS traffic are tested.

6.1. Preliminary Experiments. At the beginning of the experimental evaluation, we conduct two preliminary experiments, including parameters tuning and feature evaluation. The former is used to pick out the optimal parameters for PF. Besides, we design a simple experiment, namely, feature evaluation, to compare the proposed features and their source features. The details are specified below.

6.1.1. Parameters Tuning. As mentioned above, four parameters need to be determined in the PF implementation. We define the four parameters as m, n, t, and k. Specifically speaking, m denotes the number of iterations in the training process. n denotes the number of iterations in the fold-in process. t denotes the number of "topics". k denotes the number of nearest neighbors of KNN. To find out the optimal parameters, we devise a method by ourselves. In the following, we show our method by three specific experiments based on SSH55. Each experiment is used to determine one parameter.

To determine m and n, we need to fix the value of t and k. Furthermore, we let m equal to n according to previous PLSI applications. Then, m and n are set as several different values. Subsequently, we run the PF algorithm, draw the accuracy curve, and choose the best parameter value. Similarly, we determine the parameters t and k in the closed-world setting. The experimental results are demonstrated in Figure 7. As the left figure in Figure 7 shows, the accuracy of PF does not improve as the number of iterations rises from 50 into 150. For efficiency, we set the parameters m and n as 50. Known from the middle figure in Figure 7, the accuracy reaches maximum when the parameter k equals 1. Hence, the parameter k is set as 1 in this scenario on the SSH55 dataset. In the right figure of Figure 7, we get the maximal accuracy when the parameter t is set as 150. Naturally, we set the parameter t as 150. Similarly, we obtain the optimal parameters' values used on other datasets.

6.1.2. Feature Evaluation. We perform feature evaluation by comparing the effectiveness of the proposed new type of

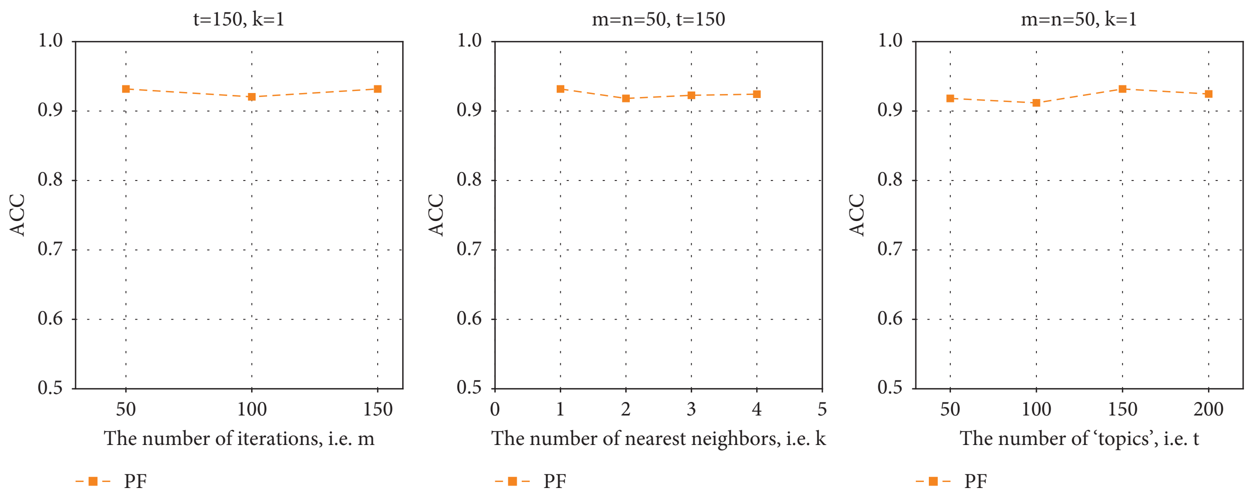


Figure 7: The results of the experiments of parameters tuning based on SSH55.

features, i.e., the "topic" probability vector, with the existing type of features, i.e., packets length with direction. For the sake of fairness, the KNN classifier is applied for both two kinds of features. For convenience, the latter attack is named LF (Length Fingerprinting). We compare PF and LF in a closed-world scenario on the AleSS73 dataset. The accuracy of PF and LF is 99.79% and 51.46%, respectively. Thus, it is evident that our proposed new type of features are more effective than the existing features. That is to say, by leveraging the PWFP model, we obtain a type of features that are more informative and powerful than the existing features, which paves a road for devising a concise traditional machine-learning attack.

6.2. Closed-World Evaluation. The closed-world evaluation includes two kinds of experiments. They are different in that whether the dataset is defended or not. For the nondefended datasets, we test six datasets based on three different PETs, including Shadowsocks, SSH, and TLS. For the defended datasets, we created twenty datasets to perform our experiments.

6.2.1. Attack on Nondefended Datasets. To validate the feasibility of a WFP attack and tune the proper parameters' values for a WFP attack, the closed-world evaluation is fundamental and critical. We leverage six nondefended datasets of various types in this closed-world scenario.

The PF attack needs to set four parameters to start. Specifically, we set the number of iterations as 50 for both the model training and fold-in process; meanwhile the number of nearest neighbors is set as 1. As for the number of "topics," we set 150 for SSH55, SSH100, HTTPS100, Alexa74, and AleSS73, while set 400 for AleSS287. It is noted that we utilize the original codes to run the state-of-the-art attacks and each algorithm is run five times to obtain its mean performance. The performance of all the attacks is shown in Table 3.

According to Table 3, the PF attack attains a stable better accuracy performance than the other attacks, even reaches an accuracy as high as 99.79% on AleSS73, while KNN and PHMM show fluctuating performance on different datasets. Specifically, PF reaches an accuracy of above 93% on five datasets, above 95% on three datasets. It beats other attacks, including DF that is based on deep neural networks, on all the datasets. As for KNN and KFP, their accuracy performance decreases to 70.91% and 43.64% on SSH55, 56.01% and 47.00% on SSH100, 78.75% and 55.75% on HTTPS100, respectively. The oscillation of performance of KNN and PHMM probably ascribes to their sensitivities to some factors, such as the number of training instances, the data quality. Although KFP achieves a comparable performance with PF on AleSS73, Alexa74, and SSH100, and DF attains a comparable performance with PF on AleSS73, they have respective disadvantages compared with PF. As for DF, when the number of training instances reduces, more numbers of iterations are needed to obtain a better accuracy, which requires more time. Regarding KFP, it needs more types of features than PF, thus introducing more feature engineering work. The results demonstrate that PF is highly effective in different scenarios, including Shadowsocks, SSH, and TLS.

Moreover, we investigate the impact of different ratios of the number of training instances to the number of total instances on classification accuracy. The results are shown in Figure 8. Since the steeper curve in Figure 8 means the more sensitive to the change of the number of training instances for each attack, it can be concluded that PF and PHMM are less sensitive to the change of training ratio than other attacks. As the results show, PF only needs rare training samples to reach a high success rate. To be specific, the PF attack only uses 4 training samples to obtain a success rate of 91.46%. This is a piece of good news for WFP, which is bothered by the data staleness issues [16]. Conversely, the accuracy curve of DF is the steepest one. Specifically, the accuracy of DF improves from 39.9% to 73.95% when the ratio increases from 20% to 40%, and continues to rise as the ratio further increases. This is in line with the characteristics of deep-learning methods.

Table 3: The results of the closed-world evaluation on nondefended datasets.

Datasets	KNN	KFP	DF	PHMM	PF
SSH55	70.91	88.37	85.45	43.64	93.18
SSH100	56.01	86.00	85.50	47.00	87.75
HTTPS100	78.75	90.00	88.25	55.75	93.5
Alexa74	96.96	97.42	93.24	97.97	98.48
AleSS73	96.57	98.97	99.24	91.51	99.79
AleSS287	82.66	82.68	86.41	62.89	95.73

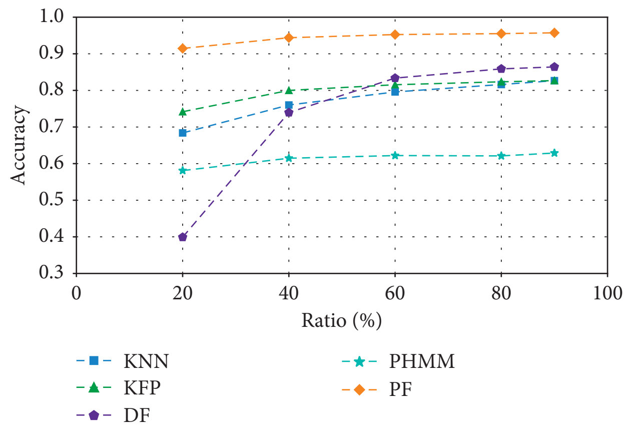


Figure 8: Closed World: The impact of different ratios of the number of training instances to the number of total instances on classification accuracy based on AleSS287.

6.2.2. Attack on Defended Datasets. In this evaluation, we consider two typical countermeasures in Shadowsocks, namely, probabilistic MTU padding defense, and probabilistic dummy packet defense, which were put forward by Zhuo et al. [13]. The former one means padding the specific packets to MTU. Whether a packet needs to be padded depends on a given probability. The latter defense denotes inserting a new packet with random size and direction following the original packet in a given probability. It is noticed that the decisions, namely, whether to pad or insert, are made by every packet in a traffic instance with the same probability. The more the probability is, the more overhead and disturbance are introduced. In total, we produce 20 defended datasets with different padding or inserting probabilities, ranging from 10% to 100%, based on AleSS287.

Note that we use the same parameters as the closed-world evaluation on nondefended datasets in this part. Table 4 shows the results of each attack against the defended datasets. Results show that our attack can well resist the two typical defenses, and achieves the best performance among all the evaluated attacks. Specifically, in the evaluation of attack on the probabilistic dummy packet defense, PF attains an accuracy of 55.66% when the inserting probability is 50%, even reaches an accuracy of 23.17% when the inserting probability becomes as high as 100%, whereas DF gets an accuracy of 34.15% and 1.92%, and PHMM obtains an accuracy of 0.53% and 0.35% in the two situations. Similarly, in the evaluation of attack on the probabilistic MTU padding defense, the accuracy of PF reaches up to 32.93% when the padding probability becomes as high as 90%, while PHMM only attains an accuracy of 0.17% by this time. It can be concluded that the two probabilistic defenses need a relatively high overhead to resist PF compared to PHMM and DF, which indicates the strong distinguishing ability of our proposed features.

6.3. Open-World Evaluation. In this part, we first investigate the impact of the parameter k on the performance indicators in the open-world setting, including TPR, FPR, and BDR. Then, two typical experiments are performed to evaluate the performance of PF.

6.3.1. Impact of the Parameter k. The parameter k is vital to the KNN algorithm. In the KNN implementation, the algorithm picks out the top k closest training samples for each test sample. Then, it assigns the test sample to the category that most of the k samples belong to. Known from the basic principle of KNN, the test sample tends to be assigned to the categories with larger sample sizes as the parameter k increases. Hence, to investigate the impact of k on the performances of PF in an open-world setting, we design this experiment. To be specific, we set all the parameters except for k as those in the closed-world setting while varying the value of k.

We perform seven experiments. The parameter k is set as 1, 3, 5, 7, 9, 13, and 21, respectively, in each experiment. Furthermore, the performance indicators, i.e., TPR, FPR, and BDR, are recorded and plotted in Figure 9. From the curves in Figure 9, FPR shows a slight decrease while BDR indicates a minor increase as k increases. In addition, the TPR curve shows a slight increase when k is less than 3, whereas it slowly declines as k increases from 3. The results are consistent with previous expectations. From the general tendency of the curves, k can be used to tune the performance of PF. For example, if the adversary puts more emphasis on TPR, k should be set as a small value. On the contrary, if FPR is more important, then k should be set as a large value.

6.3.2. Open-World Evaluation. In the open-world evaluation, there exist two kinds of training strategies for the attacker. As for the first one, the attacker not only trains the monitored dataset but also the unmonitored dataset, whereas he only trains the monitored dataset under the

Table 4: Accuracy in a closed-world scenario on the AleSS287's defended datasets.

Attack on the probabilistic dummy packet defense											
Attacks	$p = 0$	$p = 10$	$p = 20$	$p = 30$	$p = 40$	$p = 50$	$p = 60$	$p = 70$	$p = 80$	$p = 90$	$p = 100$
PF	95.73	90.67	80.65	72.99	62.53	55.66	42.59	39.55	34.05	26.22	23.17
PHMM	62.89	60.10	55.92	52.79	43.72	34.15	24.04	11.15	5.74	2.96	1.92
DF	86.41	18.03	1.22	0.53	0.53	0.53	0.53	0.35	0.35	0.35	0.35
Attack on the probabilistic MTU padding defense											
Attacks	$p = 0$	$p = 10$	$p = 20$	$p = 30$	$p = 40$	$p = 50$	$p = 60$	$p = 70$	$p = 80$	$p = 90$	$p = 100$
PF	95.73	94.68	93.99	93.03	90.06	89.72	84.64	76.04	63.93	32.93	0.78
PHMM	62.89	61.84	58.01	56.97	50.17	38.15	26.13	8.36	1.90	0.17	0

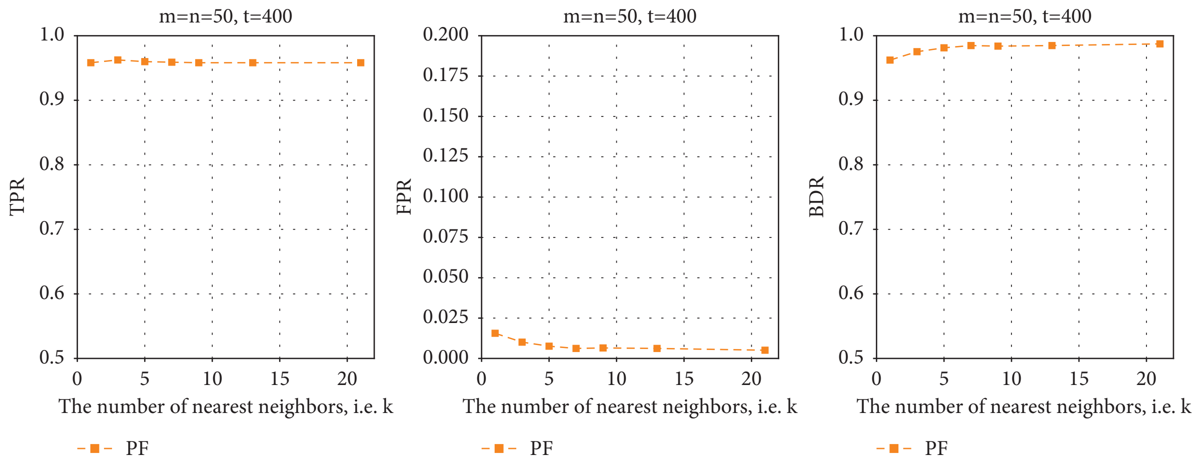


Figure 9: The impact of the parameter k on the performances of PF in an open-world setting.

second strategy. We consider the first strategy, which is called the standard model in previous literature [19]. Since PHMM takes the second strategy in the original paper, all the attacks besides PHMM are evaluated on the AleSS287 and OpwSS6879 datasets.

We conduct two experiments here. We first investigate the impact of different ratios of the number of unmonitored training instances to the number of total unmonitored instances on the performance of all the attacks, except for PHMM. In this experiment, we conduct five tests and draw the curves of TPR, FPR, and BDR for all the attacks in each test. The ratio is set as 50%, 60%, 70%, 80%, and 90% for each test. The results are shown in Figure 10. We can see that the PF attack obtains an over 89.48% TPR in all cases. The performance is superior to all other attacks. Besides, PF attains an over 93.07% BDR, a below 1% FPR in all the tests. Such performance beats all other attacks in all situations. Although KFP reaches a comparable FPR and BDR performance with PF in most tests, its TPR performance is far falling behind PF.

Next, we experiment and draw the Precision-Recall curves for comparing these attacks. Since the size of the monitored and unmonitored datasets is heavily unbalanced, the Precision-Recall curves are extensively used to represent the performance of the classifiers to avoid the base-rate fallacy [5]. At first, we take 90% of monitored instances and 80% of unmonitored instances for the training, while the rest of the instances are used for the test. Then, we conduct a set of tests by configuring a series of different settings for each attack. For different attacks, the configuring methods are different. Specifically, for KNN, KFP, and PF, we configure multiple settings by varying the parameter k following previous literature [19]. However, since DF uses the prediction probability to classify the input traffic instances, we take multiple thresholds of probability as different settings. Finally, the precision and recall performance of each attack in each test is calculated. According to the evaluation results of all the attacks, the Precision-Recall curves are plotted.

As we see in Figure 11, our attack achieves a performance of more than 97% precision and 93% recall in all settings, which outperforms the other attacks remarkably. To be specific, when the precision is 97%, the recall of DF, KFP, and KNN is remarkably less than PF. Similarly, when the recall is 93%, the precision of DF and KNN is also less than PF. For KFP, when KFP attains a comparable high precision performance (e.g., 1) as FP, its recall performance is under 80%, whereas the recall of PF is over 93%. Hence, the Precision-Recall curves demonstrate the good performance of PF.

7. Discussion

In this study, our goal is to seek another traditional machine-learning attack that merely uses one type of features, which are more effective than existing features. This goal mitigates the burden of feature engineering work for WFP. By combining the probabilistic topic model with WFP, we devise the new type of features, i.e., "topic" probability

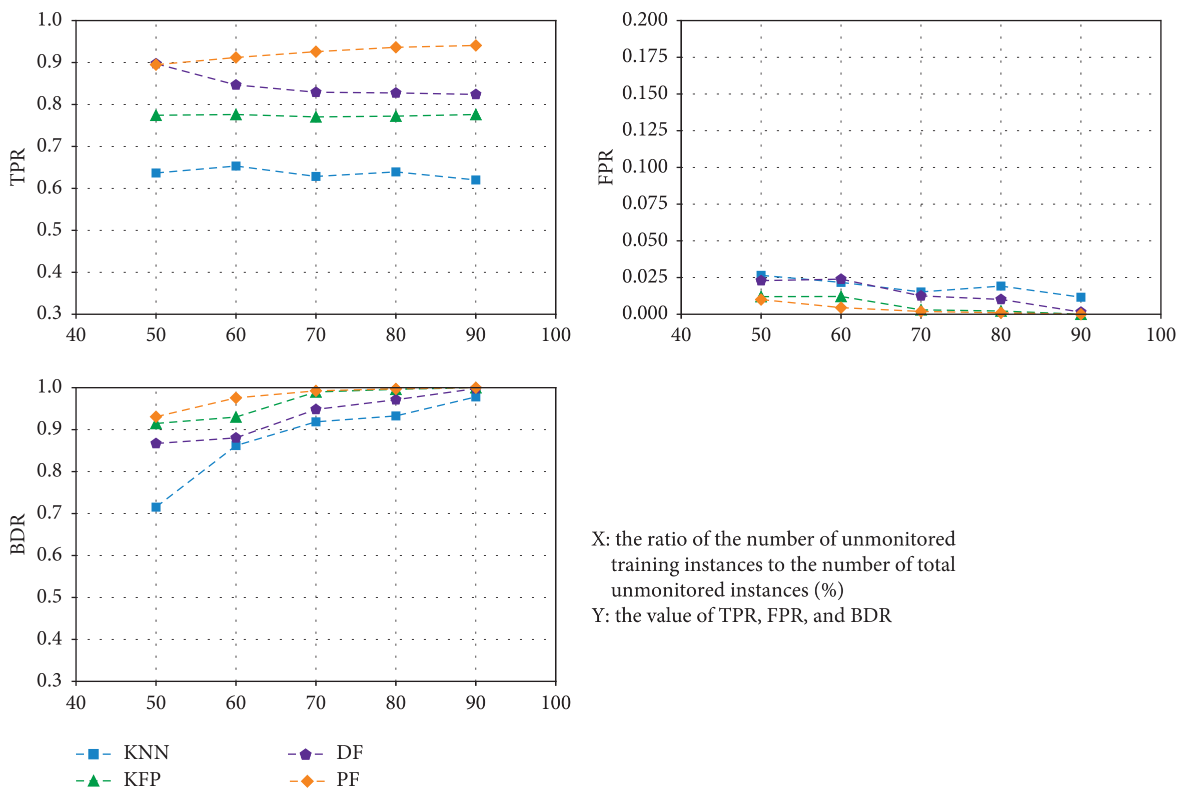


FIGURE 10: Open World: The impact of different ratios of the number of unmonitored training instances to the number of total unmonitored instances on TPR, FPR, and BDR.

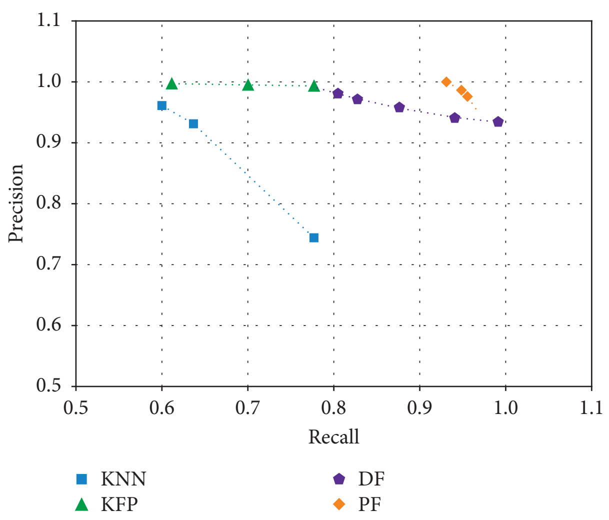


FIGURE 11: Open World: The Precision-Recall curves.

vector. Based on the new features, the PF attack is proposed and evaluated on the Shadowsocks, SSH, and TLS traffic. To be more sound and persuasive, we not only use the open datasets but also the datasets collected in a realistic scenario. The evaluation results show the good performance of PF, which proves that using a probabilistic topic model for WFP is workable.

The reason that PF is effective can be explained by the angle from which we look at a traffic instance. Traditionally, a traffic instance is represented by a length vector, each component of which indicates the value of packet length with direction. Hence, each component of the vector has the same physical significance (i.e., a length), which means that the traffic instance is merely viewed and described in a one-dimensional space. However, in the PF scheme, a traffic instance is represented by the "topic" probability vector, whose components have different physical significances that indicate the probabilities of the traffic instance belonging to different "topics." Hence, each traffic instance is mapped to a multidimensional space in our work. This truth probably explains the reason why PF is more effective than LF and performs best in both the closed-world and open-world evaluations.

Besides lessening the boring work of feature engineering, PF also reduces the number of features fed into the classifier. For example, the KNN attack needs 3736 features as its input. Besides, DF takes 5000 features as the input of DNNs. However, the needed number of features in PF equals the number of "topics," which is one of the model parameters and usually has the same order of magnitude as the number of monitored webpages (|″topics″|/|monitored web pages| < 200%) according to the experimental results.

Regarding the future work, since our key idea lies in the combination of probabilistic topic model with WFP, other models, such as LDA (latent Dirichlet allocation), HDP (hierarchical Dirichlet process), should be effective in WFP too. Note that our work gives an example of digging out a new type of distinguishing features from the existing (i.e., baseline) type of features, i.e., packets length with direction. However, the baseline type of traffic features is not limited to that used in this paper. Besides, the symbolization method can be designed in another way too. Similar work might be the potential new directions to improve and extend our attack.

Besides, our method might be used in other related fields, such as radio frequency fingerprint identification (RFFID) [37, 38]. RFFID is a lightweight access authentication method in mobile edge computing. It uses the radio frequency signal fingerprint of the wireless devices for identification. Generally speaking, the first process of RFFID is offline to establish a fingerprint database for legitimate wireless devices. Then, the fingerprint is used on the subsequent online authentication process. Like a website fingerprinting attack, if the fingerprint of wireless devices can be converted into a symbol with some rule, it is very possible to apply our method in the second process of RFFID. Such direction has great potential.

8. Conclusions

In this work, we are the first to investigate the performance of a WFP attack using the probabilistic topic model. Our work is inspired by a neglected truth, that is, the sequence file generated from a traffic instance and a plain text are similar essentially. Then, we propose the PWFP model and the PF attack. Furthermore, our attack is tested and compared with four state-of-the-art attacks. The results in three extensively applied scenarios, i.e., Shadowsocks, SSH, and TLS, prove that PF is feasible and effective. In all, we find and leverage the new type of features, i.e., the "topic" probability vector, to identify the test webpages under the protection of different PETs, and obtain a better performance than prior attacks, including a deep learning attack. To the best of our knowledge, it is the first time that a traditional machine-learning attack beats a deep-learning attack. The success of PF means that there exists great potential information for existing traffic features. Besides, it indicates that the performance of current WFP attacks might further improve while using fewer types of features and need less feature engineering work if the unexploited potential information of known traffic features is dug out and leveraged. It points at a research direction for the future.

Data Availability

To ensure the reproducibility of the evaluation results, the source code and datasets of this work will be provided upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (No. 2018YFB0204301) and the Science and Technology Innovation Plan of Hunan Province (No. 2020RC2047). The authors are very grateful to Marc Liberatore, Zhongliu Zhuo, Tao Wang, Jamie Hayes, and Payap Sirinam for their useful feedback and sharing their code and data with us, which greatly expanded the scope of our work.

References

[1] L. Max Mc, "Shadowsocks," 2016, http://www.shadowsocks.org/.

[2] J. Wu, S. Guo, H. Huang, W. Liu, and Y. Xiang, "Information and communications technologies for sustainable development goals: state-of-the-art, needs and perspectives," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2389–2406, 2018.

[3] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting," in *Proceedings of the IEEE International Conference on Cloud Computing Technology and Scienc*, pp. 31–42, IEEE, Beijing, China, December 2009.

[4] M. Liberatore and B. N. Levine, "Inferring the source of encrypted HTTP connections," in *Proceedings of the Computer and Communications Security*, pp. 255–263, ACM, Alexandria, VA, USA, October 2006.

[5] A. Panchenko, F. Lanze, A. Zinnen et al., "Website fingerprinting at internet scale," in *Proceedings of the Network and Distributed System Security Symposium*, February 2016.

[6] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the Workshop on Privacy in the Electronic Society*, pp. 103–114, ACM, Waterloo, Canada, October 2011.

[7] T. Wang and I. Goldberg, "Improved website fingerprinting on Tor," in *Proceedings of the Workshop on Privacy in the Electronic Society Bloomington*, pp. 201–212, ACM, Indiana, U.S USA, November 2013.

[8] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: website fingerprinting attacks and defenses," in *Proceedings of the Computer and Communications Security*, pp. 605–616, Association for Computing Machinery, Raleigh, NC, USA, October 2012.

[9] Y. Zhao, X. Ma, J. Li, S. Yu, and W. Li, "Revisiting website fingerprinting attacks in real-world scenarios: a case study of Shadowsocks," in *Proceedings of the International Conference on Network and System Security*, pp. 319–336, Springer, Hong Kong, China, August 2018.

[10] M. Shen, Y. Liu, L. Zhu, X. Du, and J. Hu, "Fine-grained webpage fingerprinting using only packet length information of encrypted traffic," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2046–2059, 2021.

[11] M. Shen, Y. Liu, S. Chen, L. Zhu, and Y. Zhang, "Webpage fingerprinting using only packet length information," in *Proceedings of the ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, May 2019.

[12] M. Guo, J. Fei, and Y. Meng, "Deep nearest neighbor website fingerprinting attack Technology," *Security and Communication Networks*, vol. 2021, Article ID 5399816, 14 pages, 2021.

[13] Z. Zhuo, Y. Zhang, Z.-l. Zhang, X. Zhang, and J. Zhang, "Website fingerprinting attack on anonymity networks based on profile hidden Markov model," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1081–1095, 2018.

[14] K. Abe and S. Goto, "Fingerprinting attack on Tor anonymity using deep learning," in *The Asia Pacifc Advanced Network*APAN), 2016.

[15] S. Bhat, D. Lu, A. Kwon, and S. Devadas, "Var-CNN and DynaFlow: improved attacks and defenses for website fingerprinting," 2018, https://arxiv.org/abs/1802.10215.

[16] P. Sirinam, N. Mathews, M. S. Rahman, and M. Wright, "Triplet fingerprinting: more practical and portable website fingerprinting with N-shot learning," in *Proceedings of the Computer and Communications Security*, London UK, November 2019.

[17] V. Rimmer, D. Preuveneers, M. Juarez, T. V. Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," in *Proceedings of the Network and Distributed System Security Symposium*, February 2018.

[18] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proceedings of the Usenix Security Symposium*, pp. 143–157, USENIX Association, San Diego, CA, USA, August 2014.

[19] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: undermining website fingerprinting defenses with deep learning," in *Proceedings of the 2018 Acm Sigsac Conference on Computer and Communications Security*, October 2018.

[20] M. S. Rahman, P. Sirinam, N. Matthews, K. G. Gangadhara, and M. Wright, "Tik-tok: the utility of packet timing in website fingerprinting attacks," 2019, https://arxiv.org/abs/1902.06421.

[21] H. Cheng and R. Avnur, "Traffic analysis of SSL encrypted web browsing," 1998, http://wwwcsberkeleyedu/daw/teaching/cs261-f98/projects/final-reports/ronathanheyningps 1998.

[22] Q. Sun, D. R. Simon, Y. M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu, "Statistical identification of encrypted Web browsing traffic," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 19–30, IEEE, Berkeley, CA, USA, January 2002.

[23] A. Hintz, "Fingerprinting websites using traffic analysis," in *Proceedings of the International Workshop on Privacy Enhancing Technologies*, June 2003.

[24] L. Lu, E. C. Chang, and M. C. Chan, "Website fingerprinting and identification using ordered feature sequences," in *Proceedings of the European Conference on Research in Computer Security*, September 2010.

[25] G. He, M. Yang, X. Gu, J. Luo, and Y. Ma, "A novel active website fingerprinting attack against tor anonymous system," in *Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design*, May 2014.

[26] J. Hayes and G. Danezis, "K-fingerprinting: A robust scalable website fingerprinting technique," in *Proceedings of the 25th Usenix Security Symposium*, pp. 1187–1203, USENIX Association, Austin, TX, USA, August 2016.

[27] X. Luo, P. Zhou, E. W. W. Chan, W. Lee, R. K. C. Chang, and R. Perdisci, "HTTPOS: sealing information leaks with browser-side obfuscation of encrypted flows," in *Proceedings of the Network and Distributed System Security Symposium*, pp. 1–20, San Diego, California, USA, February 2011.

[28] T. Wang and I. Goldberg, "Walkie-talkie: an efficient defense against passive website fingerprinting attacks," in *Proceedings of the 26th USENIX Security Symposium*, pp. 1375–1390, USENIX Association, Dallas, TX, USA, August 2017.

[29] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-Boo, I still see you: why efficient traffic analysis countermeasures fail," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 332–346, IEEE, San Francisco, CA, USA, May 2012.

[30] V. Shmatikov and M.-H. Wang, "Timing analysis in low-latency mix networks: attacks and defenses," in *Proceedings of the European Conference on Research in Computer Security*, pp. 18–33, Springer, Hamburg, Germany, September 2006.

[31] X. Cai, R. Nithyanand, and R. Johnson, "CS-BuFLO," in *Proceedings of the Workshop on Privacy in the Electronic Society*, November 2014.

[32] T. Wang and I. Goldberg, "Comparing website fingerprinting attacks and defenses," Technical Report 2013-30, 2013.

[33] C. V. Wright, S. E. Coull, and F. Monrose, "Traffic morphing: an efficient defense against statistical traffic analysis," in *Proceedings of the Network and Distributed System Security Symposium*, pp. 237–250, San Diego, California,C.A, USA, February 2009.

[34] S. Eddy, *HMMER User's Guide: Biological Sequence Analysis Using Profile Hidden Markov Models*, HMMER User's Guide, 1998.

[35] T. Hofmann, "Probabilistic latent semantic indexing," *International ACM SIGIR Conference on Research and Development in Information Retrieval*, vol. 51, no. 2, pp. 50–57, 1999.

[36] T.-C. Chou and M. C. Chen, "Using incremental PLSI for threshold-resilient online event analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 3, pp. 289–299, 2008.

[37] F. Xie, H. Wen, J. Wu et al., "Data augmentation for radio frequency fingerprinting via pseudo-random integration," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 3, pp. 276–286, 2019.

[38] S. Chen, H. Wen, J. Wu et al., "Radio frequency fingerprint-based intelligent mobile edge computing for internet of things authentication," *Sensors*, vol. 19, no. 16, p. 3610, 2019.