I Can Still Observe You: Flow-level Behavior Fingerprinting for Online Social Network

Yebo Feng*, Jianzhen Luo†, Chengyan Ma‡, Teng Li§, Liang Hui¶
*University of Oregon, †Guangdong Polytechnic Normal University, ‡§Xidian University,
¶China Academy of Information and Communications Technology (CAICT)
Email: *yebof@uoregon.edu, †luojz@gpnu.edu.cn

***Abstract*—The privacy of online social networks (OSNs) remains a major concern for today's Internet. Researchers have demonstrated that by analyzing inter-packet or packet-level network traffic, a third-party analyzer is able to fingerprint a user's OSN behavior information even when the traffic is encrypted. In this paper, we propose a learning-based approach that steps further to perform OSN behavior fingerprinting only through highly compressed, flow-level network traffic (e.g., NetFlow). By preprocessing flow records, segmenting traffic flows into bursts, and leveraging a long short-term memory network to classify the bursts, our approach can identify major OSN behaviors (e.g., Facebook post, Twitter Read, Weibo video, etc.) with nearly 90% accuracy. Compared with packet-level fingerprinting approaches, our approach significantly improves the fingerprinting efficiency in evaluations, making large-scale OSN usage monitoring feasible only with limited computing resources and coarse-grained network traffic. This work also reveals the huge risks facing privacy of OSN users on today's Internet today.**

***Index Terms*—traffic analysis, behavior fingerprinting, online social network, flow-level traffic monitoring**

I. Introduction

Traffic analysis (TA) is the process of examining network traffic to deduce hidden information about the ongoing communication, which can be performed even in the presence of message encryption [1]. As today's Internet messages are becoming more and more mysterious due to the increasingly large-scale use of encryption algorithms, proxy nodes, and tunneling techniques, people can hardly monitor packets by simply inspecting their payloads. TA is thus becoming a vital and indispensable approach for both network administrators and eavesdroppers to gain insights into network activities. For example, people have been widely using TA in DDoS detection [2], fingerprinting websites [3], monitoring mobile traffic [4], and quality of experience (QoE) measurement [5].

During the last decade, with the widespread use of online social networks (OSNs), such as Twitter, Facebook, and Instagram, leveraging TA to intercept and fetch users' behavior information on OSNs becomes a trend. Such a process is also called *OSN behavior fingerprinting*. Here, the behavior information includes the type of OSN service users are using, the specific activities (e.g., Facebook post, Twitter read, and Facebook chat) users are conducting on OSNs, etc. From the perspective of eavesdroppers, OSN behavior fingerprinting is a practical approach to infer private information about users, thereby helping their further malicious conducts. From the perspective of service providers, OSN behavior fingerprinting is also a scalable approach to monitor and measure ongoing OSN events.

However, OSN behavior fingerprinting is a more challenging task compared with other types of TA as it needs to deduce fine-grained behavior information simply from content-agnostic network traffic. Fortunately, with advances of AI methods, both supervised and unsupervised learning have been explored for TA, making such tasks technically feasible [6]. For example, Coull et al. [7] combine linear regression and Naïve Bayes to infer user behaviors regarding Apple iMessage; Saltaformaggio et al. [8] leverage a K-means clustering model and an SVM model to identify various user behaviors on different OSN platforms; Liu et al. [9] utilize a random forest model to differentiate behaviors related to media streaming on Facebook, Wechat, and WhatsApp.

Nevertheless, existing OSN behavior fingerprinting approaches are all based on packet-level network traffic, which requires intercepting every packet from the OSN communication to extract features. Such a process is expensive and difficult to scale. To make traffic monitoring affordable, scalable, and efficient, common practices usually aggregate relevant packets into a flow and then capture metadata or statistical information to represent that flow [6]. This is especially true for medium or large-scale networks. Therefore, migrating OSN behavior fingerprinting to flow-level environments is essential for deploying this technology at scale, even though it is quite challenging.

In this paper, we propose a flow-level OSN behavior fingerprinting approach. After preprocessing the input flow records, our approach utilizes a clustering algorithm to segment traffic into bursts, where each burst represent an OSN behavior. Then, our approach leverages an LSTM machine learning model to learn and classify the traffic burst into different OSN behaviors according to extracted features.

To our best knowledge, our approach is the first to perform OSN behavior fingerprinting on top of content-agnostic flow-level network traffic. According to evaluations, it has significant efficiency improvements compared with existing packet-level approaches, enabling a third party to directly

Corresponding author: Jianzhen Luo.

978-1-6654-3540-6/22 © 2022 IEEE

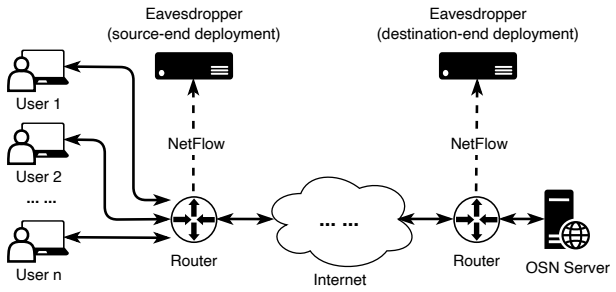


Fig. 1: The operation model of proposed approach, where two types of deployment locations are provided: source-end and destination-end.

observe all users' OSN usages in a medium-size network only with a personal computer. Meanwhile, the fingerprinting accuracy only drops by less than 10% compared to packet-level fingerprinting. This work not only offers a new efficient OSN eavesdropping paradigm, but also reveals that OSN users' privacy faces a serious breach crisis on today's internet.

II. Operation Model

In this section, we elaborate on the operation model of our approach, indicating its deployment location and data input.

A. Deployment

As our approach is based on TA, it requires inputting bidirectional (i.e., both inbound and outbound) network traffic from the monitoring network. Due to this nature, not every vantage point in the network can observe suitable traffic because in-network-based observation points usually capture asymmetric traffic due to asymmetric routing [10]. Besides, as traffic engineering approaches have been widely deployed in today's network, it is usually difficult for in-network-based observation points to capture robust, stable network traffic. Therefore, the ideal deployment location for our approach is on the gateway of the a network, either on the source-end (i.e., user-side) or destination-end (i.e., server-side). Such vantage points can observe both inbound and outbound traffic related to OSN services, which provides a solid foundation for OSN behavior fingerprinting.

Figure 1 illustrates the two operation models of our approach. To conduct analysis, our approach fetches flow-level traffic from the gateway router of either the source-end or destination-end. From the perspective of network administrators, our approach can help measure OSN service usage or investigate OSN QoE. From the perspective of attackers, this approach can be used to deduce users' private behavior information related to OSN.

B. Input data

Unlike existing approaches that need packet-level or inter-packet-level information to operate, our approach simply needs flow-level network traffic. A network traffic flow is defined as a sequence of relevant network packets sent from a source to a destination for the same application [6]. Usually, in a flow-level traffic capture system (e.g., Netflow [11], Argus [12], IPFIX [13]), the capture engines no longer copy or make snapshots of each packet, instead, they first aggregate relevant packets into a flow and then capture metadata (i.e., information from packet headers) and statistical information to represent that flow. Therefore, flow-level traffic engines are able to capture and log network traffic at large scales, but with considerable information loss.

In our implementation, we use Netflow records as the input. Our approach can also easily migrate to traffic flow environments, such as Argus flow and IPFIX. In addition, we do not need to utilize all the attributes provided by Netflow, a small set of attributes is enough to for our approach to perform OSN behavior fingerprinting. Table I lists all the attributes required by our approach and shows an example captured by Netflow.

TABLE I: Attributes we use to perform OSN behavior fingerprinting (an example captured by Netflow).

Destination IP	231.157.117.6	**Source IP**	42.127.17.43
Destination Port	443	**Source Port**	7652
Portocol	TCP	**Packet Number**	5
Bytes	856	**TCP Flag**	...A....
Start Timestampe	1582824775.808	**End Timestampe**	1582824802.688

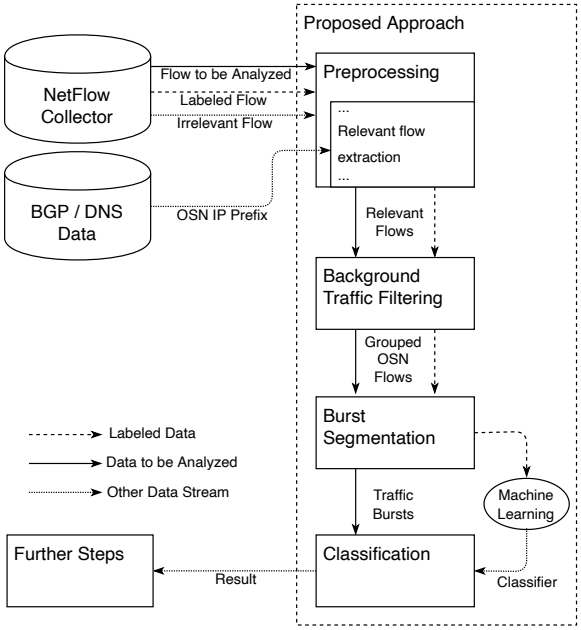


Fig. 2: The flow chart of our approach.

III. System Design

In this section, we elaborate on the design of our approach.

A. Overview

Figure 2 illustrates the flow chart of our approach. Our approach has two modes: training and inference.

Besides inputting flow-level network traffic from observation points, it also needs Border Gateway Protocol (BGP) or DNS information for reverse IP address lookup to pick out only OSN related traffic. Then, our approach filter out background traffic and preprocess relevant traffic for inference. After segmenting traffic into bursts, where each burst represents a behavior, our approach leverages machine learning algorithms to learn and classify OSN behaviors.

B. Preprocessing and preparation

The preprocessing and preparation steps mainly serve for the two purposes below:

- Comb the massive traffic flows collected from the network, grouping them by users for further inference.
- Clean the traffic flows by filtering out irrelevant flows and background flows.

During operations, the traffic capture engine will stream real-time network traffic of a network N to our approach. Such data streaming contains a set of flow records F ($F = \{f_1, f_2, ..., f_n\}$), where flow records of different IP address are mixed with each other. So the first preprocessing step is to group flow records by IP addresses according to Equation 1, thereby flows related to the same IP in the monitoring network can be in the same flow set G.

$$\begin{aligned} f_n \in G_a, \text{ iff } f_n \in F \\ \text{and } (f_n.destIP = a \text{ or } f_n.srcIP = a). \end{aligned} \tag{1}$$

Then, in each flow set G, not all traffic flows are related to OSN services. Actually, only a small proportion of network traffic are associated with OSNs. To extract only OSN-relevant traffic, we check each flow's both destination and source IP addresses. If one of these IPs is belonging to an OSN, we consider that this flow is directly related to an OSN. We will record the start timestamps and end timestamps of the flow, and treat all the flows P within this time window as potentially OSN-relevant. Here, to obtain accurate OSN IP prefixes, we perform reverse IP addresses lookup from a DNS server near the network or directly fetch a group of OSN IP prefixes using BGPStream API [14].

Of a particular note is that flows inside P are not all OSN flows, actually they contain three sets of flows: uncorrelated background traffic flows B (i.e., network traffic sent at the time but has nothing to do with OSN services), traffic flows R that directly related to an OSN (i.e., network traffic sent from or to an OSN server), and traffic flows I that indirectly related to an OSN (i.e., network traffic sent for an OSN service but to or from an non-OSN IP address). So our next preprocessing target is to clean the data stream by removing B from P. This is challenging because it is difficult to differentiate between flows in set B and I. We observed that background traffic usually have a longer duration and happen before OSN flow appears. Therefore, our approach will keep buffering a couple seconds of traffic for each IP addresses inside the monitoring network before any OSN flow appears. This set of traffic is denoted as Bu. Whenever an OSN IP prefix appears and a flow set P is extracted, our approach will treat all the flows that appeared in both Bu and P as B. Our approach will remove such background flows from P (as Equation 2 shows)

$$\begin{aligned} O &= R \cup I \\ &= \{f | f \in P \text{ and } f \notin Bu\} \end{aligned} \tag{2}$$

and pass the remaining traffic flow O for further processing .

C. Burst segmentation

By previous steps, we can already obtain only OSN-related traffic flows. However, the data we have now are still sets of disordered flow records, where different behaviors' traffic are mixed together. We thus need to segment the flows into bursts, where each burst represents an OSN behavior conducted by the user. Then we can make inferences on each specific traffic burst, rather than dealing with miscellaneous flows generated by different OSN behaviors.

To design an appropriate traffic segmentation algorithm, we conducted a simple measurement on daily traffic related to OSNs. We found that each OSN behavior (e.g., send messages, post images, read post) will immediately incur several outbound and inbound network packets for necessary data transmission. From the perspective of traffic flows, the packet density and byte density will increase with the occurrence of each OSN behavior. Thus, by separating bursts of traffic flows, we can separate traffic flows for different OSN behaviors.

We utilize density-based clustering to segment traffic bursts. More specifically, our burst segmentation is similar to Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [15], by which two density thresholds are taken into account to find density-connected areas. Our approach examines both length and byte density to identify the gap between two traffic bursts, thereby segmenting traffic flows to represent OSN behaviors.

The burst segmentation procedure works as follow:

1) Placing flows in O on time coordinates, divide the duration of each flow f into multiple time bins of equal length and define a flow point d_t^f for each time bin t.
2) Derive time series data D ($D = [b_{t1}, b_{t2}, ..., b_{tn}]$) from O using Equation 3.

$$b_{tn} = \sum_{f \in O \text{ and } tn \subset f.time} d_{tn}^f.bytes \tag{3}$$

3) Calculate the segmentation timestamps with Algorithm 1, where we identify all the low-density and long gaps to determine traffic bursts.
4) Generate traffic bursts by dividing and aggregating traffic flows with the calculated timestamps.

Our approach then transmits the generated bursts to the next step for behavior inferences.

D. Behavior classification

In this step, our approach inputs traffic bursts generated from the previous steps. Then, it extracts features from the traffic bursts and classifies them into different OSN behaviors using machine learning.

Algorithm 1 Burst segmentation with density-based clustering

input: time series data D
input: time threshold δ and byte threshold ϵ
$time\_count \longleftarrow 0$, $results \longleftarrow [\,]$ ▷ parameter initialization
for i in D **do**
  if $i \leqslant \epsilon$ **then**
    $time\_count \longleftarrow time\_count + 1$
  else if $time\_count \geqslant \delta$ **then**
    append the index number of i to $results$
    $time\_count \longleftarrow 0$
  else
    $time\_count \longleftarrow 0$
  end if
end for
convert the indexes in $results$ to timestamps
return $results$

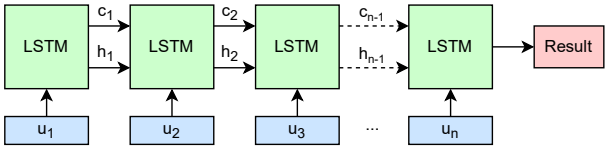


Fig. 3: Architecture of the LSTM classification model.

Just like the previous segmentation step, our approach places flows in a traffic burst on a time coordinate and divide the duration of the burst into multiple units (denoted as u) of equal length. Then, our approach extracts a set of features from each unit, including the number of bytes, number of flows, number of packets, number of different TCP flag types, ratio of inbound bytes to outbound bytes, ratio of inbound packets to outbound packets. This feature set covers the majority of information we can derive from flow-level traffic data and is compatible with all the widely-used flow-level traffic capture engines (e.g., Netflow, sFlow, Argus, etc.).

Once features are extracted, we leverage a Long Short Term Memory (LSTM) network [16] to perform the OSN behavior classification. As a special type of recurrent neural network (RNN), LSTM is good at learning hidden knowledge from time series data in a long-term manner, which fits our OSN behavior fingerprinting task. Additionally, LSTM is easy to converge without the need to carefully fine-tune the parameters.

Figure 3 illustrates the architecture of our LSTM classification model. Our approach keeps streaming traffic burst units to the LSTM model until the end of the behavior. The LSTM model will output the classification result after the last burst unit, indicating which OSN behavior category the current traffic burst belongs to. In our implementation, we set a single hidden layer with 15 neurons for our LSTM model as the input data is not complicated.

IV. Evaluation

In this section, we evaluate our approach from multiple aspects, including the parameter setup, classification efficacy, deployability, and operation efficiency.

A. Dataset

As our approach is learning-based, it requires labeled data to train the classification model and perform evaluation. We thus utilized a synthetic dataset consisting of a part of a public dataset and a self-collected dataset in this project. The public dataset is the USTC-TFC2016 dataset [17], which consists of some network traffic traces related to a few OSN sites in PCAP format. However, as the dataset only contains a limited number of OSN sites (e.g., Weibo) and OSN behaviors and is not well-labeled, we also captured around 3.25 GB of packet-level labeled OSN network traffic related to Twitter, Facebook, and Weibo in our experimental network. The experimental network consisted of 15 devices, including both mobile devices and personal computers. After cleaning, the final dataset contains 5540 labeled OSN behaviors. The types of OSN behaviors are listed in Table II. We used 90% of the data to train the classification model and 10% of the data to perform the tests. The testing dataset did not engage in the training process.

B. Setup

Although the input datasets we used are in PCAP formats, we transferred them to NetFlow records before inputting them to our approach. We also directly used the raw PCAP data to compare the performances of flow-level and packet-level OSN behavior fingerprintings. The training and testing tasks were performed on a personal computer with an Intel i7 8700k CPU, an Nvidia GTX 1080 GPU, and a 32-GB memory.

For the NetFlow record generation, we set the active flow timeout as 3 seconds, which is a small time frame that can reserve informative, fine-grained traffic information but simultaneously maintain decent system performance. As for the burst segmentation, according to our experiments, we set the byte threshold ϵ as 5 because a small amount of background traffic (e.g., packets for notifications, TCP control packets) still exist between two burst and this number ensures that the burst segmentation will not be disturbed by such background traffic.

To determine the suitable setup for the time threshold δ, we tried different δ values and recorded the corresponding segmentation results' purity scores. Here, the purity score indicates how accurate the segmentation is, which is widely-used in clustering assessments. Figure 4 illustrates the trend of purity scores. We can see that the purity score can reach the peak when the δ is around 1.6. Therefore, we set δ as 1.6 in our approach.

C. Efficacy of OSN behavior fingerprinting

We evaluated our approach's efficacy of OSN behavior fingerprinting using the test dataset. Table II lists the detailed evaluation results, including the number of ground truth, false positive rates, recall scores, and precision scores for each type

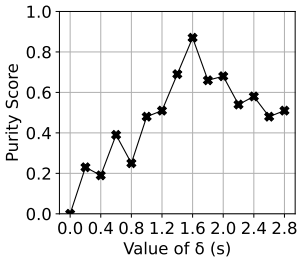


Fig. 4: Segmentation purity scores with different time threshold δ values.

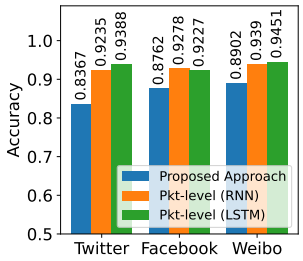
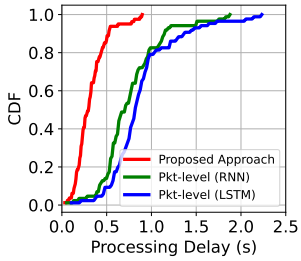
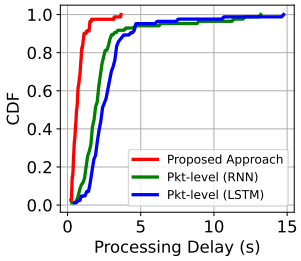


Fig. 5: Accuracy gaps between flow-level and packet-level approaches.



(a) 50 packets per second.



(b) 500 packets per second.

Fig. 6: Processing delay comparisons.

TABLE II: Efficacy of OSN behavior fingerprinting (FPR: false positive rate, REC: recall, PRE: precision).

OSN Behavior	Number of Ground Truth	TP	FP	FPR	REC	PRE
Twitter Read (content loading)	55	49	5	0.0100	0.8909	0.9074
Twitter Post (with image)	21	21	0	0.0000	1.0000	1.0000
Twitter Post (without image)	39	28	12	0.0233	0.7179	0.7000
Twitter Video	41	40	1	0.0019	0.9756	0.9756
Twitter DM Chat	40	26	14	0.0272	0.6500	0.6500
Facebook Read (content loading)	67	53	12	0.0246	0.7910	0.8154
Facebook Post (with image)	37	36	8	0.0155	0.9730	0.8182
Facebook Post (without image)	50	41	4	0.0079	0.8200	0.9111
Facebook Video	40	40	0	0.0000	1.0000	1.0000
Weibo Read (content loading)	68	57	10	0.0206	0.8382	0.8507
Weibo Post (with image)	25	25	0	0.0000	1.0000	1.0000
Weibo Post (without image)	41	34	8	0.0156	0.8293	0.8095
Weibo Video	30	30	0	0.0000	1.0000	1.0000

of OSN behavior studied. Our approach is able to reach an 0.8664 F1 score in OSN behavior fingerprinting. In particular, it performs better when identifying network traffic related to media transmission (i.e., image and video).

D. Comparison with packet-level approach

As features generated from packets are of finer granularity, packet-level fingerprinting approaches should have better efficacy compared to flow-level approaches. We thus compared our approach with packet-level approaches to study how large the gap is. We utilized two packet-level approaches with RNN and LSTM algorithms. The input features were in the same type. The only difference was that packet-level approaches input features per packet, rather than per time unit as the proposed flow-level approach does. Figure 5 illustrates the comparison results. We can see that the accuracy differences are not large—only less than 10%. Considering the information loss for flow-level data and the efficiency improvements, such a result is acceptable.

E. Operation efficiency

In the end, we evaluated and compared the operation efficiency of our approach by testing the processing delay with different volumes of traffic inputs. Figure 6 illustrates the cumulative distribution functions (CDFs) of our approach and the two packet-level approaches when inputting 50 pkts/s and 500 pkts/s of traffic data. From the results, we can see that benefiting from the low system overhead of flow-level traffic processing, our approach significantly outperforms packet-level approaches in terms of processing times, enabling it to processing a large amount of traffic at the line speed.

V. Related Work

TA has been used for multiple types of tasks by both attackers and network administrators [18]. Recently, with advances of AI-based methods, the application scope and efficacy of TA have been significantly enhanced. By capturing and analyzing the network traffic data, even in encrypted forms, an analyzer is able to detect ongoing DDoS attack [2], perform QoE investigation [5], infer the website the users are visiting [19], measure usages of different web services [20], or even infer the location of users [21].

Among different categories of TA tasks, behavior fingerprinting for OSN is particularly challenging because it tries to deduce fine-grained application-layer information out of encrypted traffic data, which requires deep understandings about how traffic patterns change with different application-layer behaviors [6]. Moreover, as users' behaviors are unpredictable on OSNs and there are many different OSN behaviors (e.g., content reading, liking, posting, uploading, etc.), it is almost impossible to perform OSN behavior classification only based on a small amount of rules.

Back in 2009, Schneider et al. [22] measured and studied OSN usages from the perspective of network traffic for four different OSN sites. They found that different OSN behaviors will incur different statistical characteristics of network traffic. Later, researchers have proposed several approaches to leverage machine learning to identify different OSN behaviors. For example, Coull et al. [7] combined linear regression, naïve bayes, and rule lookup table to classify user behaviors on messaging applications; Fu et al. [23] utilized a hidden Markov model to differentiate behaviors on Wechat and WhatsApp;

Liu et al. [9] leveraged K-means clustering and a random forest classifier to identify different medium sharing behaviors on multiple OSN sites; some researchers [8], [24] also fingerprinted more types of OSN behaviors by collecting and studying more traffic data.

However, all the aforementioned approaches are based on packet-level or inter-packet-level network traffic, which is limited for the following reasons: (1) packet-level traffic capture is not scalable considering the capacity of current Internet infrastructures. Conversely, flow-level traffic capture is efficient and widely deployed, which only incurs less than 5% overhead for most routers and switches [6]; (2) packet-level traffic data is not easily accessible for both network administrators and attackers; (3) as the information contained in flow-level traffic data is largely reduced, the feature extraction and classification steps can be much more efficient and can easily reach the line speed.

Therefore, although flow-level OSN behavior fingerprinting is more challenging considering the information loss and compression, the benefits it brings are significant. Actually, a few research projects have been focusing on studying flow-level OSN traffic [25], but they are either not aimed at identifying OSN behaviors or weak in this type of classification tasks.

VI. Conclusion

In this paper, we propose a learning-based approach to extend OSN behavior fingerprinting from packet-level to flow-level. After segmenting network flows into bursts to represent user behaviors on OSNs, our approach utilizes an LSTM model to classify each burst into a specific OSN behavior, thereby measuring OSN usages, performing OSN QoE investigations, or even eavesdropping users' private activity information. Compared to existing packet-level approaches that usually incur high system overheads, the proposed approach is efficient and does not have strict requirements on the input data. With the computing of a personal computer, our approach can easily monitor all the OSN behaviors within a company-level network in line speed and with acceptable accuracy. This work also reveals the huge risks facing privacy of OSN users on the Internet today.

Acknowledgments

This research is funded by the special project in key fields of Guangdong Universities (2021ZDZX1031), National Natural Science Foundation of China (No. 61902291), the founding of Shaanxi Key Laboratory for Network Computing and Security Technology (NCST2021YB-03), and the Fundamental Research Funds for the Central Universities (XJS211516).

References

[1] R. Soltani, D. Goeckel, D. Towsley, and A. Houmansadr, "Towards provably invisible network flow fingerprints," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2017, pp. 258–262.

[2] Y. Feng and J. Li, "Toward explainable and adaptable detection and classification of distributed denial-of-service attacks," in *International Workshop on Deployable Machine Learning for Security Defense*. Springer, 2020, pp. 105–121.

[3] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 1187–1203.

[4] D. Naboulsi, M. Fiore, S. Ribot, and R. Stanica, "Large-scale mobile traffic analysis: a survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 124–161, 2015.

[5] M. H. Mazhar and Z. Shafiq, "Real-time video quality of experience monitoring for https and quic," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 1331–1339.

[6] Y. Feng, "Toward finer granularity analysis of network traffic," Area Exam, 3 2022, available at https://www.cs.uoregon.edu/Reports/AREA-202203-Feng.pdf.

[7] S. E. Coull and K. P. Dyer, "Traffic analysis of encrypted messaging services: Apple imessage and beyond," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 5–11, 2014.

[8] B. Saltaformaggio, H. Choi, K. Johnson, Y. Kwon, Q. Zhang, X. Zhang, D. Xu, and J. Qian, "Eavesdropping on {Fine-Grained} user activities within smartphone apps over encrypted network traffic," in *10th USENIX Workshop on Offensive Technologies (WOOT 16)*, 2016.

[9] J. Liu, Y. Fu, J. Ming, Y. Ren, L. Sun, and H. Xiong, "Effective and real-time in-app activity analysis in encrypted internet traffic streams," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 335–344.

[10] M. Crotti, F. Gringoli, and L. Salgarelli, "Impact of asymmetric routing on statistical traffic classification," in *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*. IEEE, 2009, pp. 1–8.

[11] B. Claise, "Cisco systems netflow services export version 9," Tech. Rep., 2004.

[12] "Argus project," https://openargus.org/, date of visit: 2021-11-25.

[13] B. Claise, M. Fullmer, P. Calato, and R. Penno, "Ipfix protocol specification," *Interrnet-draft, work in progress*, 2005.

[14] C. Orsini, A. King, D. Giordano, V. Giotsas, and A. Dainotti, "Bgpstream: a software framework for live and historical bgp data analysis," in *Proceedings of the 2016 Internet Measurement Conference*, 2016, pp. 429–444.

[15] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] W. Wang and D. Lu, "The ustc-tfc2016 dataset," https://github.com/yungshenglu/USTC-TFC2016, 2016, date of visit: 2021-10-15.

[18] E. Papadogiannaki and S. Ioannidis, "A survey on encrypted network traffic analysis applications, techniques, and countermeasures," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–35, 2021.

[19] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 605–616.

[20] J. Mirkovic, Y. Feng, and J. Li, "Measuring changes in regional network traffic due to covid-19 stay-at-home measures," *arXiv preprint arXiv:2203.00742*, 2022.

[21] I. Trestian, S. Ranjan, A. Kuzmanovic, and A. Nucci, "Measuring serendipity: connecting people, locations and interests in a mobile 3g network," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, 2009, pp. 267–279.

[22] F. Schneider, A. Feldmann, B. Krishnamurthy, and W. Willinger, "Understanding online social network usage from a network perspective," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, 2009, pp. 35–48.

[23] Y. Fu, H. Xiong, X. Lu, J. Yang, and C. Chen, "Service usage classification with encrypted internet traffic in mobile messaging apps," *IEEE Transactions on Mobile Computing*, vol. 15, no. 11, pp. 2851–2864, 2016.

[24] E. Papadogiannaki, C. Halevidis, P. Akritidis, and L. Koromilas, "Otter: A scalable high-resolution encrypted traffic identification engine," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2018, pp. 315–334.

[25] Y. Feng, J. Li, L. Jiao, and X. Wu, "Towards learning-based, content-agnostic detection of social bot traffic," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2149–2163, 2020.