Snowflake Anonymous Network Traffic Identification

Yuying Wang, Guilong Yang, Dawei Xu(✉), Cheng Dai, Tianxin Chen, and Yunfan Yang

College of Cybersecurity, Changchun University, Changchun, China
xudw@ccu.edu.cn

Abstract. Tor, as a widely used anonymous communication system, is frequently employed by some users for illegal activities. Snowflake server as a plugin that enables users to connecting to the Tor network, allowing users to evade surveillance by connecting to the Tor network through it. Since Snowflake hides user traffic within regular WebRTC, it becomes challenging for authorities to differentiate and regulate, posing significant difficulties in monitoring efforts. To address these issues, this paper proposes a feature extraction method based on traffic statistical characteristics and a Snowflake traffic identification model based on MLP. We collected traffic datasets in Docker environment, extracted variable-length DTLS handshake sequences, and employed the feature extraction method to extract their statistical characteristics, including packet length, session duration, and average time between sending two packets, among other features. The MLP-based Snowflake traffic identification model can determine whether the traffic belongs to the target traffic based on these features. Moreover, this method can accurately identify traffic even when the traffic fields change. Experimental results demonstrate that this method achieves a 99.83% accuracy rate in identifying Snowflake traffic. Additionally, even when the data distribution in the dataset is altered, although the method requires more training iterations, it still achieves a 99.67% accuracy rate.

Keywords: Anonymous communication · Tor network · Traffic identification · MLP · Deep learning

1 Background

With the rapid development of the Internet, people's awareness of privacy protection has grown stronger. Various anonymous communication systems have become increasingly popular. According to the CNNIC report on March 2, 2023, as of December 2022, China's online population reached 1.067 billion, with an Internet penetration rate of 75.6%, surpassing over half of the country's population. With such a large user base, anonymous communication systems are often abused by some malicious users, leading to serious cybersecurity incidents [1]. As the most widely anonymous communication system, the Tor system allows users to anonymously access various websites through three-hop relays, without being detected by service providers or intermediaries. The

Tor network has millions of users, and some individuals exploit its anonymity for illicit activities such as drug and firearm trafficking. Researchers have employed traffic identification techniques to identify Tor traffic and block it. In response, the Tor development team introduced Pluggable Transport (PT) [2] technology, which utilizes clients with special protocols as the first hop in the relay chain. These protocols can hide special traffic within normal traffic, making it difficult for censors to discern. Among the most widely used are Obfs4, Meek, and Snowflake. Snowflake primarily evades censorship by concealing traffic within the data channel of WebRTC [3]. WebRTC utilizes DTLS [4] as the underlying protocol for data transmission. Being a widely adopted audio and video technology, WebRTC poses significant challenges to censorship efforts. This paper focuses on identifying the DTLS handshake process within WebRTC. It collects target traffic within Docker environment and employs a feature extraction method based on traffic statistical characteristics to extract features. Finally, a Snowflake traffic identification model based on Multi-Layer Perceptron (MLP) is utilized to determine whether the traffic belongs to the target traffic. If it does, the censoring authority can choose to block the traffic; if not, the traffic is allowed to pass through.

2 Related Research

Snowflake [5] is currently one of the most widely used PTs. David et al. [6] conducted a study by collecting a significant amount of application traffic that utilizes WebRTC communication. They manually analyzed the differences between these traffic samples and identified the potential to recognize Snowflake based on certain fields. Additionally, they highlighted the use of WebRTC as a means to evade censorship. Kyle MacMillan et al. [7] conducted a study to evaluate the recognizability of Snowflake. By collecting multiple application traffic samples that utilize WebRTC technology at the underlying level, they analyzed the interaction packets and protocol fields during the DTLS handshake phase. The researchers proposed a method to identify Snowflake traffic based on the DTLS handshake fields. Building upon the research conducted by MacMillan et al., Chen et al. [8] expanded their dataset and proposed a framework for Snowflake traffic identification. This framework utilizes rule matching and DTLS fingerprinting to determine whether user traffic is accessing the Tor network and further classify whether the user is accessing hidden services within the Tor network.

The earlier research primarily focused on the data transmission phase of Snowflake, where WebRTC utilizes the DTLS protocol for initial connection establishment. By extracting specific fields from the protocol and transforming them into features, machine learning algorithms were employed to identify and differentiate between normal WebRTC traffic and abnormal traffic. They primarily focused on the fixed data packets in the DTLS handshake, specifically the Client Hello and Server Hello. However, this approach faces a significant challenge when Snowflake modifies these fields to make them identical to normal WebRTC fields. In such cases, it becomes difficult to distinguish between the two using this method.

3 The Fundamental Principles of Snowflake

As a pluggable transport plugin in Tor, Snowflake operates differently from the original Tor network. In the Tor-Meek plugin, the client primarily utilizes a technique called domain fronting [9]. It involves initially connecting to a Content Delivery Network (CDN) provider that supports this technique. The CDN provider then forwards all of the client's traffic to the next hop node. In this scenario, all client traffic accessing the anonymous system needs to be relayed through the CDN provider, resulting in significant bandwidth overhead.

Snowflake as a new PT consists of the following components, as illustrated in Fig. 1 of the system architecture. The client will first send a proxy request to the broker, and then the broker will create an SDP answer containing information about the proxy and respond to the client. At this point, the client will directly connect to the proxy. When the proxy receives a connection from the client, it first checks if the received request is legitimate, meaning it matches the information it previously sent to the broker. If the information matches, the proxy accepts the client's connection. The client initiates the DTLS handshake by sending the necessary information to generate a session key. Once the key is generated, both the client and the proxy use it to exchange messages over the WebRTC data channel.

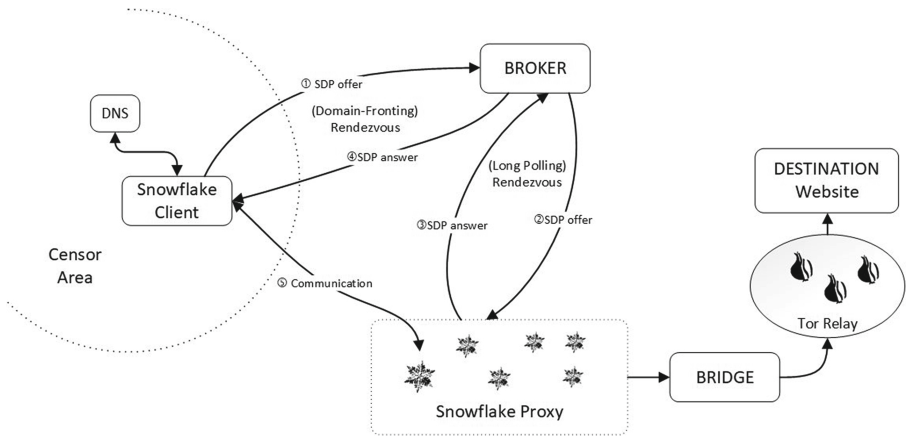


Fig. 1. Snowflake system architecture diagram.

4 MLP-Based Snowflake Traffic Identification

4.1 Feature Extraction

This paper proposes a method of feature extraction from variable-length traffic sequences using traffic statistical features, primarily based on session-level traffic. Each session-level traffic can be represented as:

$$X_n = \{x_1, x_2, x_3, \ldots, x_k\} \quad (1)$$

here, x_k represents a UDP packet, and X_n represents all communication traffic between a server and a client, with bidirectional direction.

Since this study focuses on identifying target traffic based on the handshake traffic features between the client and the proxy in WebRTC, the first step is to ensure the complete collection of all handshake traffic. The underlying protocol used for data transmission in WebRTC is DTLS. In a normal DTLS handshake protocol, the client first sends a Client Hello request to the proxy. The proxy then responds with a Server Hello, including its authentication information, such as its public key. After receiving the authentication information from the proxy, the client uses the received public key to encrypt the subsequent information and sends a request. Upon receiving the request, the proxy responds with another data packet. From there, the client can engage in regular encrypted communication with the proxy. This communication process is similar to the communication process in TLS. Under normal circumstances, the handshake traffic between the client and the proxy can be completed with only four data packets. However, due to the diversity of network environments and the use of UDP protocol at the underlying level, which does not guarantee transmission quality, in practical network scenarios, multiple retransmissions are often required. Therefore, capturing only the first four packets of the transmission phase is far from sufficient. It is necessary to capture variable-length traffic sequences from the transmission flow, which can be represented as follows:

$$S_k = \{x_1, x_2, x_3, \ldots, x_i\} \tag{2}$$

x_i represents a DTLS handshake data packet, and S_k represents all the DTLS handshake data packets.

After technical analysis and comparing a large number of data packets, it was determined that when the payload data of the first packet in UDP is 22, the current traffic corresponds to handshake traffic. The traffic that matches this feature is extracted and merged for further feature extraction.

In the traffic feature extraction section, this study primarily analyzes the target traffic using statistical methods. In this regard, we utilized the CICFlowMeter, a feature extraction software, and made certain modifications to it. This software is capable of extracting features from both TCP and UDP traffic. However, in our study, WebRTC utilizes only UDP packets. Therefore, we removed the TCP functionality from the software and incorporated the standard deviation of packet lengths, denoted as:

$$PLS = \sqrt{\frac{\sum_{i=0}^{n}(Li - \sum_{i=0}^{n} Li/n)^2}{N}} \tag{3}$$

where Li represents the length of an individual packet and N represents the number of packets, this feature represents the magnitude of variations in packet sizes during a session. The average length feature of the packets is represented as:

$$PLM = \frac{\sum_{i=0}^{n} Li}{N} \tag{4}$$

The average time feature between the forward transmission of two packets is represented as:

$$FIM = \frac{\sum_{i=0}^{n}(T_{i+1} - T_i)}{N} \tag{5}$$

where T_i represents the time at which the current packet is sent, and N represents the number of packets. This feature represents the speed of information exchange between the two parties in the session. Additionally, there are other less important features such as data statistics in the reverse flow and session duration, which are not listed here in detail.

Due to the issue of packet retransmission caused by network conditions, we have added the feature of the number of retransmitted packets. This feature can be used to indicate the quality of the network link. If the value is high, it suggests the presence of blocking nodes in the link. The feature extraction algorithm is shown in Table 1. After extracting features from DTLS session traffic, focusing on aspects such as packet size and average transmission rate, a total of 48 flow statistical features are obtained, which can be used for subsequent model training.

Table 1. Retransmission packet calculation.

Algorithm 1: Number characteristic of retransmitted packets

Input: **Session sequence**

Output: **Number of retransmission packets**

```
number = 0
packets = []
for packet in sessions:
    if packet in packets:
        number += 1
    else:
        packets.append(packet)
```

4.2 Model Introduction

Firstly, we need to preprocess the data by normalizing the data. The overall architecture of the model is illustrated in Fig. 2. The first layer is the input layer, which consists of 48 neurons. After preprocessing the data, the dimensionality of the feature vector is 48. The second layer consists of 512 neurons. The input vector dimension is 48, and the output vector dimension is 512. The third layer consists of 64 neurons. The input vector dimension is 512, and the output vector dimension is 64. The fourth layer consists of a single neuron. It is followed by a Sigmoid function. After applying the Sigmoid function, the final output is mapped to the range [0, 1], representing the probability of the input being the target traffic. When the output probability is greater than 0.5, the model classifies the traffic as Snowflake traffic. If the probability is less than 0.5, the traffic is classified as normal WebRTC traffic.

The loss function used in this case is BCELoss, which stands for Binary Cross Entropy Loss. It is a commonly used loss function for binary classification tasks. It measures the difference between the model's output and the true labels. There are many

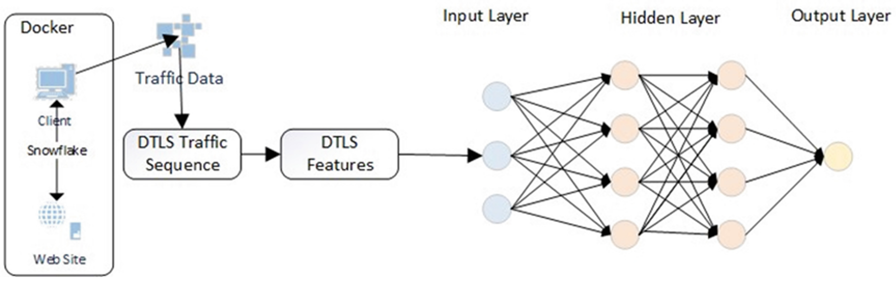


Fig. 2. Model architecture diagram.

optimization functions available, we have opted to use the Adam algorithm. Adam is an adaptive optimization algorithm that adjusts the learning rate for each parameter. It allows parameters with smaller gradients to have larger learning rates, while parameters with larger gradients have smaller learning rates. This helps improve training efficiency and speed.

5 Experiment

5.1 Collection and Processing of Traffic

Since Docker provides a closed environment and allows images to be saved in the cloud for easy distribution, the experiments in this study were primarily conducted using Docker. The base image used for the experiments was Ubuntu 20.

The first step is to download the base image and instantiate a container from it. Inside the container, you will install the Python environment, Scapy, and Selenium libraries. Then, you'll install a headless browser that does not require a graphical interface. Next, you'll install the Tor software and Snowflake. Since Snowflake is written in Go, you'll need to install the Go compiler first in order to run the client. Write a script that first uses the Scapy library to listen to the network traffic on a specific network interface. Then, use the Selenium library to write an automation script that allows the browser to visit certain websites automatically. Specify that the traffic should pass through a designated Snowflake proxy port. Finally, when the client enters the anonymous communication network through the proxy, establish a three-hop circuit to access random websites and generate traffic. After the visit, automatically save the traffic data based on the date. To streamline the process and reduce manual efforts, we utilize the Cron software to schedule the program for daily execution. The image can be found at xinbigworld/ubuntu:1.2. Once the image is downloaded, it can be directly run to obtain the aforementioned container environment.

After running the program on two servers for a certain period of time, we consolidate all the Pcap files and merge them into a single file. We then utilize a feature extraction method based on traffic statistics to extract features from the traffic. The extracted features are directly saved to a text file for easy access during the subsequent model training.

By leveraging data from previous papers and combining it with the traffic collected in our Docker environment, we obtained a dataset of size 6477. In this dataset, Snowflake represents the target traffic, while the traffic collected from the other three software represents normal traffic. Our goal is to achieve high accuracy in identifying the target traffic within this dataset. The distribution of the collected traffic data is shown in Table 2.

Table 2. Dataset distribution.

	Snowflake	Facebook messenger	Google hangouts	Discord
Sessions	1386	1584	1539	1968

The term Sessions represents a complete conversation between the client and the server, encompassing bidirectional traffic. It includes the traffic sent from the client to the server as well as the traffic sent back from the server to the client. For example, 1386 indicates that the traffic collection program was executed 1386 times, resulting in the collection of 1386 instances of DTLS handshake information.

The proposed method was utilized to extract DTLS handshake data from a large volume of traffic. The experiments were conducted on a personal computer with an Intel i5-12500H 3.1GHz CPU and 16GB of RAM. Experimental results show that extracting handshake data from a session of length 1383 takes approximately 11ms, with complete packet content including all DTLS handshake information. This demonstrates the feasibility of the proposed method in terms of both efficiency and accuracy. In terms of feature extraction, experiments conducted on a session sequence of length 192 took a total of 381ms for feature extraction.

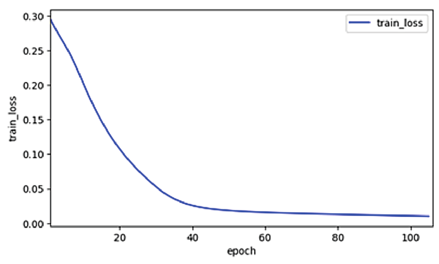
5.2 MLP Model Training

The dataset is divided into training and testing sets in a 7:3 ratio. The training set is used to train the model by updating its parameters to establish a classification model. The testing set is used to evaluate the performance of the model and assess its discriminatory power. In this section, we will use accuracy as a metric to measure the model performance.

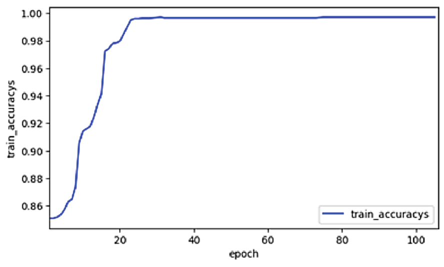
As shown in Fig. 3a, the loss value of the model keeps decreasing as the number of training iterations increases. After 40 training iterations, the loss value reaches a plateau and shows little further improvement. From Fig. 3b, it can be observed that the accuracy of the model on the training set reaches its highest value after 25 training iterations, which is 99.72% for the subsequent iterations. From Fig. 3c, it can be observed that the training performance on the testing set is similar to that on the training set. Both achieve the highest accuracy of 99.83% after 25 training iterations and show little improvement thereafter. This experiment confirms that the proposed method of feature extraction based on flow statistics and the MLP-based Snowflake traffic identification model can effectively recognize the target traffic.

5.3 Model Comparison

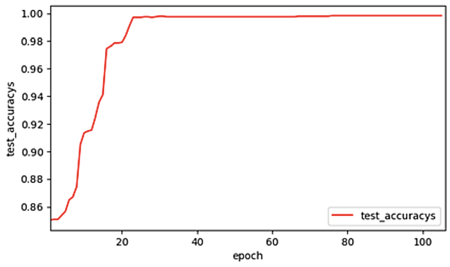
We trained four different models using the same dataset, which was divided into training and testing sets in a 7:3 ratio. We also introduced three additional performance metrics



(a) The loss value of the model



(b) The accuracy of the model on the training set



(c) The accuracy of the model on the test set

Fig. 3. Training and evaluation of models.

to evaluate the models. Among them, precision represents the proportion of correct predictions among all positive predictions. Recall represents the proportion of correct predictions among all positive instances. F-score primarily balances precision and recall. The final results are shown in Table 3. From the table, we can see that, in terms of Accuracy, all models except RF achieve relatively high values. However, in terms of precision, MLP significantly outperforms the other models, indicating that the predicted target traffic consists mostly of true target traffic. SVM is able to recall all positive samples in terms of recall, but due to its lower precision, it indicates that there will be a certain number of negative samples predicted as positive, resulting in a higher rate of false positives. In summary, the MLP model demonstrates excellent performance in all aspects of Snowflake traffic identification. Therefore, choosing the MLP model for identifying target traffic is more suitable.

Table 3. Model performance comparison

Classifier	Accuracy	Precision	Recall	F1
SVM	0.9855	0.9115	1.0	0.9537
Random Forest	0.9586	0.7892	0.9640	0.8679
MLP	0.9983	0.9916	0.9972	0.9944
Naive Bayes	0.9838	0.9078	0.9925	0.9483

5.4 Comparison of the Importance of Traffic Statistical Features

In the experiment, we can use Random Forest (RF) to rank the importance of features and obtain their weights, as shown in Fig. 4. We can observe that the most important feature accounts for 27% of the weight, representing the total time between two forward packets. The second feature occupies 22% of the weight, representing the average length of packets throughout the entire conversation. The third feature represents the duration of the session, i.e., the total time spent by the client and server on DTLS. The cumulative importance of the top three features reaches around 50%. This method involves collecting session packets during the DTLS handshake and extracting statistical features from these packets. Compared to the research presented in the second section, where they directly compare specific fields of the protocol, our method exhibits better robustness. When the protocol fields of the traffic are altered, their method may not achieve the same high accuracy. However, since our method does not rely on using protocol fields as features, it can still effectively identify the target traffic even if the protocol fields are changed.

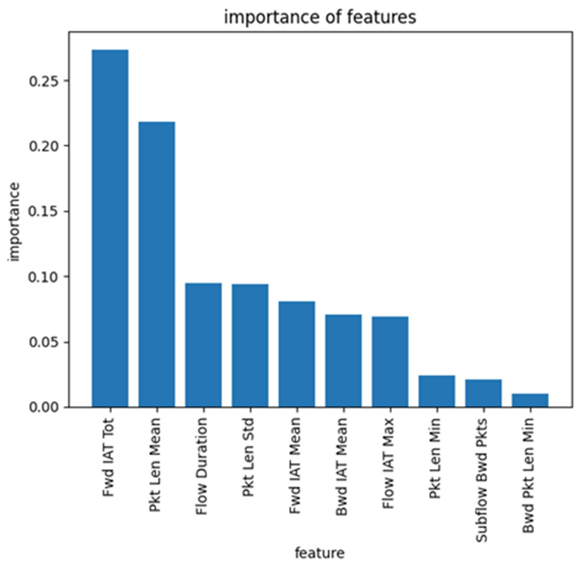


Fig. 4. Feature importance.

5.5 The Impact of Data Distribution on Model Performance

As shown in Fig. 5, by varying the proportion of target traffic in the dataset, we simulate the data distribution in real-world scenarios. For example, a ratio of 5:1 represents the size comparison between normal traffic and target traffic datasets. We can observe that the proposed feature extraction method and model still have a high probability of feasibility in real scenarios, achieving an accuracy of 99.67%.

From the figure, we can observe that when the proportion of target traffic in the dataset is higher, the model does not initially achieve a high accuracy. This is because the dataset is larger, and the model needs multiple iterations to learn the characteristics of the traffic. On the other hand, when the proportion of target traffic decreases in the dataset, although the model initially achieves a high accuracy, it requires more training iterations to reach an accuracy of 99.67%.

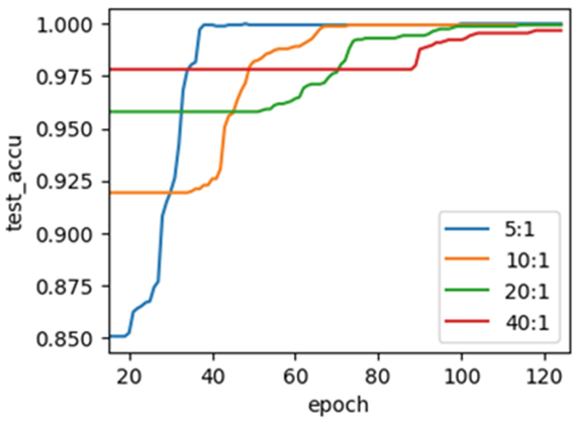


Fig. 5. Target traffic identification with different proportions.

6 Conclusion

Many users employ the Tor anonymous communication system to conceal their online activities for the purpose of engaging in illegal activities. While there are numerous methods to identify raw Tor traffic, the difficulty of traffic identification has increased significantly with the use of Tor PT technology, particularly the adoption of Snowflake. In this study, we extract 48 statistical features from DTLS traffic using a flow-based statistical feature extraction method. These features are preprocessed to obtain properly formatted input features in accordance with established standards. The extracted feature data is fed into an MLP model, which can ultimately determine whether the traffic belongs to the target flow. Even if certain fields in the traffic change, this method can still function properly because it primarily relies on the statistical features of the traffic for classification, and changes in protocol fields do not affect its recognition accuracy. Traffic identification is a dynamic process, and as we identify traffic features, Snowflake developers may modify these features to render our model ineffective. This could lead to an ongoing cat-and-mouse situation. It is necessary to continually collect traffic data to maintain a high level of accuracy. In the future, we hope to automate this process to adapt to updates and changes in Snowflake versions.

References

1. Yannikos, Y., Heeger, J., Steinebach, M.: Scraping and analyzing data of a large darknet marketplace. J. Cyber Secur. Mob. 161–186 (2023)
2. Shahbar, K., Zincir-Heywood, A.N.: Traffic flow analysis of tor pluggable transports. In: 2015 11th International Conference on Network and Service Management, pp. 178–181 (2015)
3. Sredojev, B., Samardzija, D., Posarac, D.: WebRTC technology overview and signaling solution design and implementation. In: 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, pp. 1006–1009 (2015)
4. Shaheen, S.H., Yousaf, M.: Security analysis of dtls structure and its application to secure multicast communication. In: 2014 12th International Conference on Frontiers of Information Technology, pp. 165–168 (2014)
5. The Snowflake, https://trac.torproject.org/projects/tor/wiki/doc/Snowflake, last accessed 2023/4/24
6. Fifield, D., Epner, M.G.: Fingerprintability of WebRTC. arXiv preprint arXiv:1605.08805 (2016)
7. MacMillan, K., Holland, J., Mittal, P.: Evaluating snowflake as an indistinguishable censorship circumvention tool. arXiv preprint arXiv:2008.03254 (2020)
8. Chen, J., Cheng, G., Mei, H.: F-ACCUMUL: a protocol fingerprint and accumulative payload length sample-based tor-snowflake traffic-identifying framework. Appl. Sci. **13**(1), 622 (2023)
9. Fifield, D., Lan, C., Hynes, R., Wegmann, P., Paxson, V.: Blocking-resistant communication through domain fronting. Proc. Priv. Enhanc. Technol **2015**(2), 46–64 (2015)