frontiers | Frontiers in Computer Science

TYPE Original Research
PUBLISHED 17 July 2025
DOI 10.3389/fcomp.2025.1518128

OPEN ACCESS

EDITED BY
David Aldabass,
Nottingham Trent University, United Kingdom

REVIEWED BY
Yongquan Fu,
National University of Defense Technology, China
Maikel Lázaro Pérez Gort,
Ca' Foscari University of Venice, Italy
Jorge E. López De Vergara,
Autonomous University of Madrid, Spain

*CORRESPONDENCE
Yulong Wang
✉ wangyulong@caep.cn

RECEIVED 28 October 2024
ACCEPTED 18 June 2025
PUBLISHED 17 July 2025

CITATION
Ye K, Zeng T, Duan Y, Han J, Zhong G, Chen Z and Wang Y (2025) Obfuscated malicious traffic detection based on data enhancement. *Front. Comput. Sci.* 7:1518128. doi: 10.3389/fcomp.2025.1518128

Obfuscated malicious traffic detection based on data enhancement

Ke Ye, Tao Zeng, Yubing Duan, Jun Han, Guoxin Zhong, Zhi Chen and Yulong Wang*

Institute of Computer Application, China Academy of Engineering Physics, Mianyang, China

As the proportion of encrypted traffic increases, it becomes increasingly challenging for network attacks to be discovered. Although existing methods combine unencrypted statistical features, e.g., average packet length, with machine learning algorithms to achieve encrypted malicious traffic detection, it is difficult to escape the influence of artificially forged noise, e.g., adding dummy packets. In this study, we propose a novel encrypted malicious traffic detection method named RobustDetector (RD) for obfuscated malicious traffic detection. The core of the proposed method is to use the dropout mechanism to simulate the process of original features being disturbed. By introducing noise during the training phase, the robustness of the model is improved. To validate the effectiveness of RobustDetector, we conducted extensive experiments using public datasets. Our results demonstrate that RobustDetector achieves an average F1-score of 90.63% even when random noise is introduced to the original traffic with a probability of 50%. This performance underscores the potential of our proposed method in addressing the challenges of obfuscated malicious traffic detection.

KEYWORDS

network anomaly detection, obfuscated malicious traffic detection, encrypted traffic classification, network attack and defense, deep learning

1 Introduction

With the continuous development of network technology, more services are connected to the Internet, bringing convenience to transportation, medical care, education, entertainment, etc. Meanwhile, the exponential growth of network scale is also accompanied by the proliferation of cyber attacks. The traditional cyber attack detection methods attempt to find features in the traffic packet payload, e.g., special strings that only exist in malicious traffic. However, with the development of Secure Socket Layer/Transport Layer Security (SSL/TLS), the packet payload becomes invisible, which ensures the security of the information transmission process. However, it not only brings privacy to users but also increases concealment for attackers. According to the Zscaler ThreatLabz 2023 Ransomware Report (Zscaler, 2024), 85.9% of threats are delivered over encrypted channels, and encryption malware has become one of the most serious threats to enterprise organizations, accounting for 78.1% of all observed attacks.

To solve the problems of encrypted malicious traffic detection, existing works (Arora et al., 2014; Zhang et al., 2018; Meghdouri et al., 2018; Ahmed et al., 2022; Xu et al., 2020) attempt to extract traffic features from packet headers (e.g., TCP payload length, timestamps, and TCP flags) and combine these features with machine learning algorithms to build classifiers for encrypted traffic detection. However, the method based on fixed

features may be bypassed by attackers, who obfuscate the original traffic by adding manual noises. For example, they may modify the header field of packets, adding random time latency, injecting dummy packets, etc. This obfuscated traffic can significantly reduce the effectiveness of existing methods (Liu et al., 2022).

In this study, we propose *RoubustDetector*, a novel encrypted malicious traffic detection method. We convert the raw traffic into sequence features, i.e., packet length, packet direction, and packet inter-arrival-time sequences. The sequences are regarded as the input of the model, which consists of a feature extraction module and a clustering module. The feature extraction module includes two CNN models with identical structures, the only difference being that one of the CNN models has enabled the random dropout. The identical sample is input into two models simultaneously, and the output of the two models is regarded as the features of the sample. The feature extracted through the dropout simulates the process of attackers destroying the original features randomly. The clustering module clusters the output of the feature extraction module according to the label, and the distance between each feature and its cluster center point serves as the loss of the feature extraction module. In the detection process, we confirm the category of traffic by calculating the distance between features and each cluster center point.

We summarize our contributions as follows:

- We propose the RobustDetector, a novel encrypted malicious traffic detection method for evasion attack detection. Through the feature extraction module, we use features disrupted by artificial noise to simulate obfuscated traffic. Through the clustering module, we calculate the loss to reduce the distance between original traffic and obfuscated traffic features, which enhances the robustness of the model, making it effective in obfuscated malicious traffic detection.
- We simulate various ways in which attackers add noise, i.e., adding random time delays, injecting dummy packets, modifying the port number, and adding random payloads. By regulating the probability of noise addition, we generated encrypted malicious traffic with varying obfuscation degrees.
- Using the representative public dataset CICIDS2017 (Sharafaldin et al., 2018), we conduct thorough experiments. From the results of the experiment, RobustDetector can achieve an average F1-score of 90.63% in the detection of evasion attacks, which indicates that it is robust in the classification of obfuscated traffic.

The rest of this article is organized as follows. We introduce the threat model and related work in Section 2. Then, we present the proposed RobustDetector in Section 3. Next, we conduct extensive experiments to evaluate the performance of RobustDetector in Section 4. Finally, we discuss the limitations and conclude this article in Section 5.

2 Threat model and related work

In this section, we first present the threat model for evasion attack detection, followed by a review of related work in encrypted malicious traffic detection.

2.1 Threat model

There are usually two factions in the scenario of malicious traffic detection, i.e., the detector and the attacker, as shown in Figure 1.

The attacker launches attacks on one or more devices in a local area network. The detector can collect attack traffic on the gateway or terminal devices. To avoid detection, the attack traffic is usually encrypted by the attacker, and the detector cannot obtain the payload in plain text (Shen et al., 2022). To further confuse the detector, the attacker may add random artificial noise to the original traffic (Liu et al., 2022). We assume that artificial noise includes the following types, i.e., adding random time delays, injecting dummy packets, modifying the port number, and adding random payloads, which will be introduced in detail in Section 3.4.

The attacker can randomly select both the method and parameters for noise injection, while the detector has neither prior knowledge of the noise generation strategy nor the ability to anticipate where noise will be inserted.

2.2 Related work

Prior research has investigated various machine learning approaches for malicious traffic detection, including Decision Tree (DT), Random Forest (RF), K-Nearest Neighbor (k-NN), and Support Vector Machine (SVM).

Chen et al. (2018) proposed a mobile malware detection method using highly imbalanced network traffic. The six statistical features are extracted from downlink and uplink traffic metrics, including the number of bytes in bidirectional flows. By leveraging multiple machine learning algorithms (i.e., SVM, C4.5, Gradient Boosting, Na"ive Bayes, BayesNet, and Adaboost) across varying imbalance ratios, they demonstrate malicious traffic detection performance on imbalanced datasets. Lashkari et al. (2017) collected real smartphone traffic to construct a malware traffic dataset. Their detection framework employs statistical features and machine learning, utilizing feature selection techniques (i.e., Information Gain and CFS Subset) combined with SVM to identify optimal feature subsets. The selected features are then integrated with classifiers, such as RF, DT, and k-NN, for malware identification. Rahmat et al. (2019) developed an ensemble framework using bagging and boosting. The approach partitions datasets into subsets, trains base learners, and aggregates predictions through model voting. The experiments compare five ensemble learners with four single learners, and the results show that XGBoost achieves the highest accuracy. Zhao and Ma (2024) proposed a tree-based model for fine-grained unknown attack detection in network traffic. They leverage isolation forest techniques to split the data distribution hyperplane, enabling fine-grained classification of both known and unknown attacks. It also supports incremental model updates to adapt to changing legitimate traffic through growing and retiring mechanisms, demonstrating superior performance in dynamic environments.

Traditional machine learning methods face limitations due to traffic feature diversity, particularly their reliance on manually engineered features requiring domain expertise. This constraint

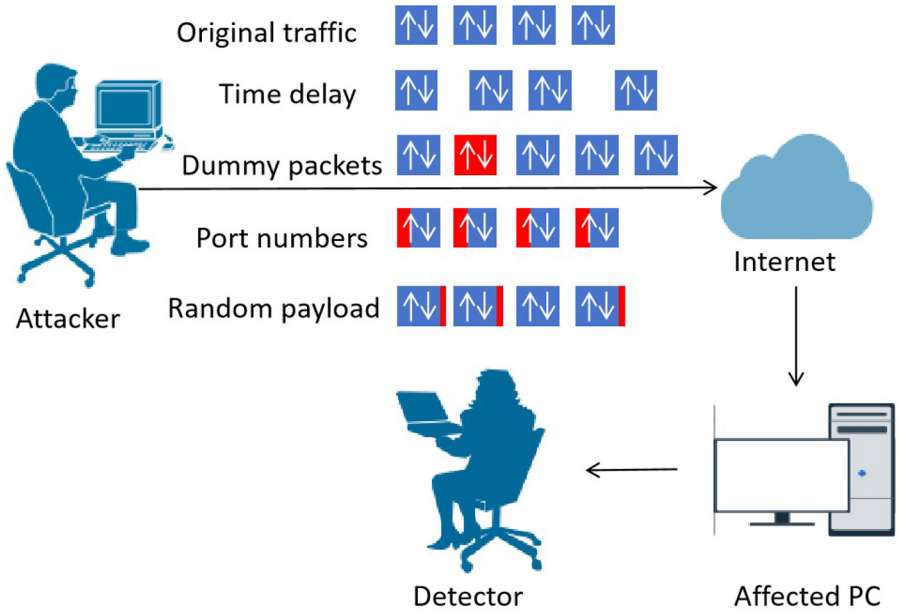

FIGURE 1
The threat model for traffic anomaly detection.

motivates the adoption of deep learning for automated feature extraction in encrypted malicious traffic detection.

Bader et al. (2022) introduced MalDIST, extending encrypted traffic classification to malware detection. Based on Aceto et al. (2021), they extracted 14 statistical features from the first 32 packets as input for a hybrid LSTM-CNN model. From the results of the experiment, the proposed method performs better than the baseline methods. Fu et al. (2022) proposed ST-Graph, a multi-stream framework that leverages multiple features from spatial and temporal perspectives for encrypted malware traffic detection. They design an attribute-heterogeneous graph to capture traffic features, and demonstrate that ST-Graph achieves superior performance in encrypted malware detection. Han et al. (2022) proposed a lightweight encrypted malware traffic detection method using autoencoders and *k*-means. Building on the existing feature set (Anderson and McGrew, 2017), they extract six types of statistical features, including packet size, TCP window size, and inter-packet timing. To enhance detection accuracy, an autoencoder is employed for feature compression, followed by *k*-means clustering on the refined features. Their unsupervised approach outperforms supervised methods, such as SVM, XGBoost, and LSTM, in experimental evaluations. Fallah and Bidgoly (2022) proposed an Android malware detection method utilizing LSTM networks. They extracted a feature vector containing 75 elements, which is fed into the model for automated feature optimization. Experimental results indicate that this deep learning-based approach achieves higher accuracy in classifying malicious software families compared to traditional methods. Yang and Lv (2022) proposed an enhanced intrusion detection system (IDS) for Internet of Things (IoT) networks by integrating deep learning and knowledge graph techniques to address the limitations of existing IDS solutions. The primary focus is on improving the detection of malicious activities through feature extraction and semantic relationship analysis among network traffic features. Jung (2024) proposed an enhanced encrypted traffic analysis method leveraging Graph Neural Networks (GNNs) and optimized feature dimensionality reduction. It classifies malicious network traffic by analyzing key features, such as IP address, port, CipherSuite, MessageLen, and JA3, within TLS session data. Shi (2025) presented an in-network malicious traffic detection system that supports continual adaptation to emerging attacks. They employ Supervised Mixture Prototypical Learning (SMPL) to learn class prototypes and transform them into priority-guided matching rules for efficient deployment on programmable switches. Zhou and Xu (2025) proposed an efficient pre-trained model for traffic data analysis named Traffic-Former. They leverage unsupervised pre-training to learn fundamental traffic semantics from unlabeled data and incorporate a fine-grained multi-classification task during pre-training to capture packet direction and order information, enhancing traffic representation. They employ a data augmentation technique, Random Initialization Field Augmentation (RIFA), to focus on key information by randomizing certain packet fields.

Although existing malicious traffic detection methods achieve high accuracy in experimental settings, their effectiveness remains unverified against evasion attacks where original traffic is perturbed by artificial noise. According to the experimental results of Liu et al. (2022), both traditional machine learning models (e.g., LR, SVM, and DT) and deep learning architectures (e.g., 1D CNN and 2D CNN) exhibit vulnerability to adversarial perturbations such as destination port modification and junk data insertion. In their study, five types of noise were systematically injected, leading to accuracy degradation across all tested methods. To address this limitation, we propose RobustDetector, a novel framework for evasion attack detection in malicious traffic analysis.

3 The proposed RobustDetector

In this section, we first present the system overview of RobustDetector and then describe the design details.

3.1 System overview

To address the challenge of detecting obfuscated malicious traffic, we enhanced model robustness by simulating noise injection during training. Specifically, dropout layers are employed to randomly mask original traffic features, mimicking attackers' random feature corruption. We propose a loss function that simultaneously minimizes intra-class feature distances and maximizes inter-class feature separation. The overall framework, illustrated in Figure 2, comprises three core components, i.e., traffic representation module, feature extraction module, and clustering module.

For traffic representation, we convert the first N packets of traffic into sequence features. The packet length, packet direction, and packet inter-arrival-time are extracted as features. In the sequence feature, we do not consider packet header fields, as these features are easily altered.

At the heart of the framework lies the feature extraction module, comprising two structurally similar CNN networks with critical functional differences. Compared to the first CNN network, the second model introduces a strategically placed dropout layer that randomly nullifies neuron activations during forward propagation, i.e., setting their output to 0. This architecture enables systematic simulation of feature corruption.

The clustering module calculates cluster centroids for each class and computes sample-to-center displacements, which are formulated as a joint loss function to train the feature extraction module.

3.2 Feature extraction module

Dropout is proposed by Hinton et al. (2012). When a complex feedforward neural network is trained on a small dataset, it is prone to overfitting. To prevent this, the performance of neural networks can be improved by preventing the collective action of neurons. Dropout can be seen as a bagging method. During the training phase, some neurons are hidden with a random probability, and during testing, all neurons are used. It is like multiple base models working together to obtain a prediction result. The forward propagation process with dropout is shown in Figure 3.

In other scenarios, dropout can also be seen as data augmentation. Due to the absence of some neurons during forward propagation, some local features will not be reflected in the output feature vector. This is like adding random noise, hiding some information from the original sample. Adding noise is commonly used to generate data augmentation samples. Dropout achieves a similar effect to adding noise by masking partial features. In contrastive learning, the distance between the original sample and the data augmentation sample is calculated as the loss.

For each traffic flow, we only truncate its first N packets and convert them to sequence features, containing packet length, packet direction, and packet inter-arrival-time sequences. The sequence features are regarded as the input of the feature extraction module. In the module, we leverage dropout to generate an obfuscated traffic feature vector. The feature extraction module contains two CNN networks with similar structures, named model 1 and model 2. The structure of the CNN network is shown in Figure 4. Compared to model 1, model 2 introduces a dropout layer (probability $= 0.4$) before the fully connected layer. This design aims to disrupt the original features as extensively as possible within a controlled range to enhance model robustness, while avoiding excessive distortion of traffic feature semantics that could lead to the learning of degenerate patterns. Notably, since the training methodology requires output consistency between the two models, over-destruction of features in model 2 may degrade the feature similarity in outputs relative to model 1, ultimately resulting in ineffective model training.

3.3 Clustering module

Considering the design goal of the framework, in the clustering module, we need to make the output distances between model 1 and model 2 closer for samples of the same class, and farther for samples of different classes. Cosine similarity is used to measure the distance, which is defined in Equation 1:

$$dis(a,b) = \frac{< a, b >}{||a|| \cdot ||b||} \tag{1}$$

where a is the output of model 1 and b is the output of model 2, $< a, b >$ represents the dot product of two vectors, $||a|| = \sqrt{a_1^2 + a_2^2 + ... + a_n^2}$.

The loss of the framework is divided into two parts, i.e., within-class distance and between-class distance. The within-class distance is defined in Equation 2.

$$WD = \sum_{l=1}^{L} \sum_{i=1}^{N_l} \sum_{j=1, j\neq i}^{N_l} dis(i,j) \tag{2}$$

where L is the number of classes, N_l is the number of samples in class l. The between-class distance is defined in Equation 3.

$$BD = \sum_{i=1}^{L} \sum_{j=1, j\neq i}^{L} dis(C_j, C_j) \tag{3}$$

where C_i is the center point of class i. We design a loss function (Equation 4) to make the feature vectors of samples in the same class closer and those of different classes farther apart.

$$loss = WD + \frac{1}{BD} \tag{4}$$

We update the model parameters of the feature extraction module to reduce the loss gradually and record the information of the cluster center points in each training epoch. The

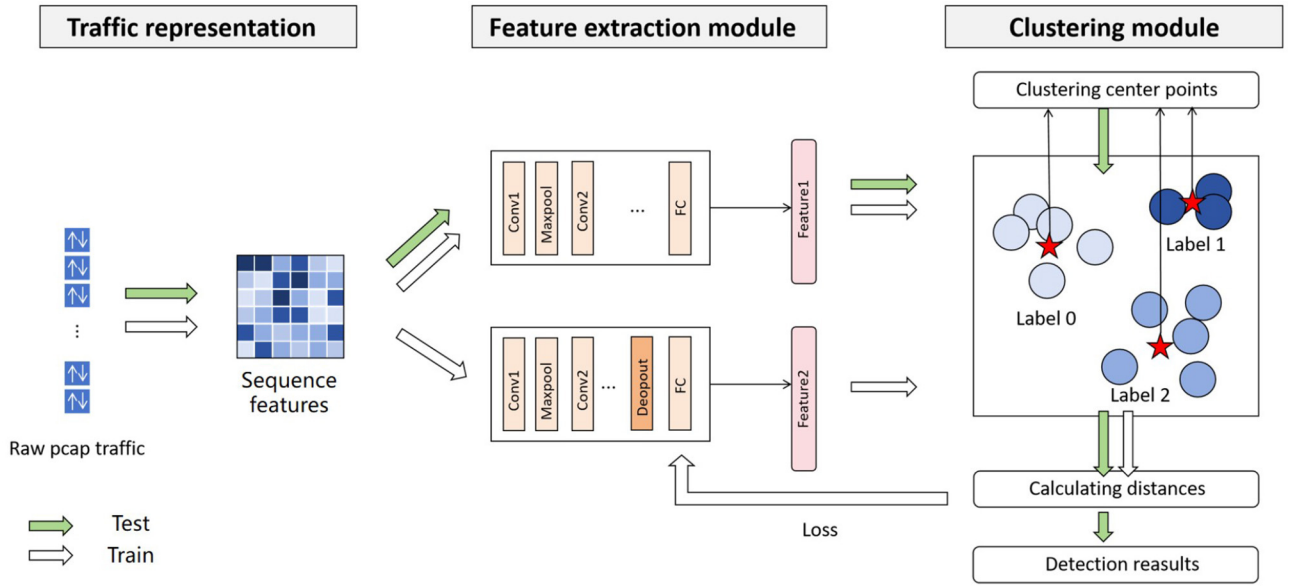

FIGURE 2
The system overview of RobusetDetector.

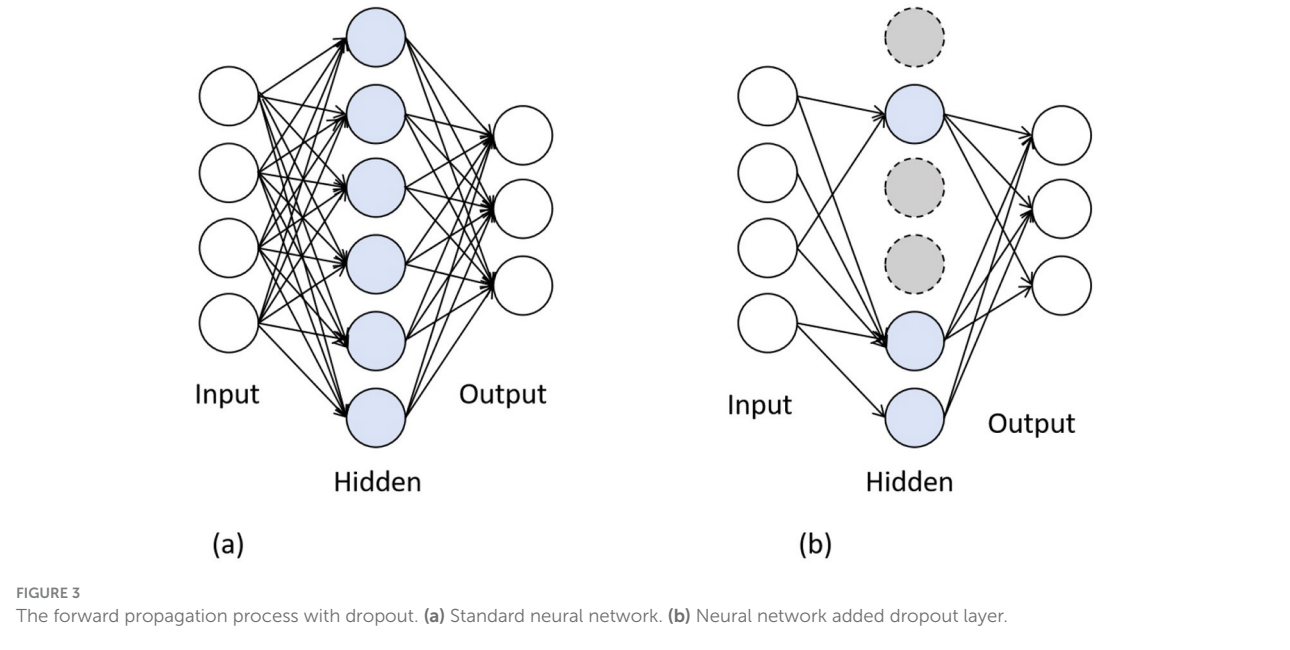

FIGURE 3
The forward propagation process with dropout. **(a)** Standard neural network. **(b)** Neural network added dropout layer.

features of the original traffic extracted by the feature extraction module become closer to those of the obfuscated traffic, achieving detection of obfuscated malicious traffic. In testing, we only use model 1 for feature extraction. The output feature vectors are compared with the cluster center points recorded during the training process. The label of the cluster center point closest to the feature vector is considered to be the detection value.

3.4 Obfuscated malicious traffic generation

To prove the effectiveness of the framework, in addition to simulating random noise through the dropout layer, we also generate real obfuscated traffic samples. We assume that the artificial noise includes four types, as these are commonly used features in existing encrypted traffic analysis works (Liu et al.,

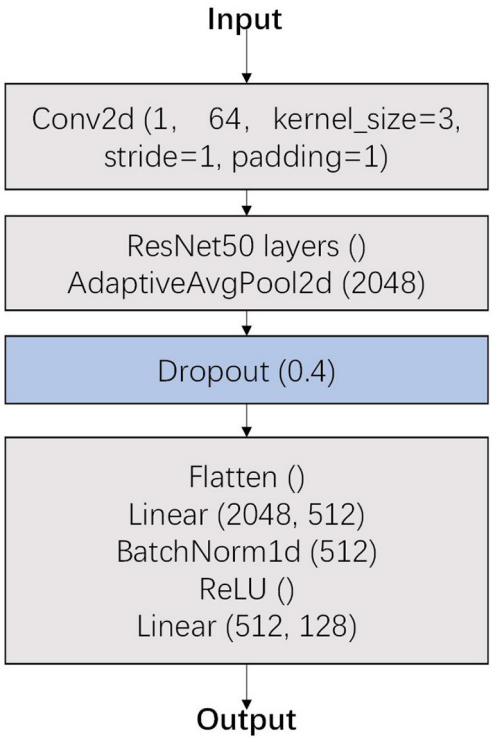

FIGURE 4
The forward propagation process with dropout.

2022). We will provide a detailed introduction to the methods for adding each type of noise.

3.4.1 Adding random time delay

For the attacker, adding random delay is one of the most covert forms of artificial noise. As even without the interference of attackers, the inter-arrival time of packets can easily change due to network conditions, making this noise difficult to perceive. However, when the attacker adds randomness, the time delay must follow the condition that each packet needs to be sent in its original order; otherwise, it may cause the receiving end to receive packets out of order. Under this assumption, we provide an algorithm for adding a random time delay in Algorithm 1 with probabilistic execution control. The noise intensity parameter $a\%$ specifies that each packet independently faces an $a\%$ probability of being delayed. If the packet is not the terminal unit in the sequence, its added delay must adhere to a strict temporal constraint, i.e., the manipulated arrival time cannot exceed the timestamp of its immediate successor packet. For the terminating packet, we impose an upper bound of 1 s on artificial delay injection to maintain protocol-compliant timeout thresholds.

3.4.2 Injecting dummy packets

Dummy packet injection serves as an active countermeasure against traffic analysis by perturbing multiple discriminative features, including packet sizes and directions. Our implementation

Input: A flow $T = \{p_1, \cdots, p_n\}$, the probability of noise generation $a\%$.
Output: The obfuscated flow T'
$T' = [\,]$
for $i \in N$ **do**
  $R = randit(1, 10)$
  if $R \leq 10 * a\%$ **then**
    if $i \neq N$ **then**
      $t = randit(0, p_{i+1}.timestamp - p_i.timestamp)$
      $p_i.timestamp = p_i.timestamp + t$
    else
      $t = randit(0, 1)$
      $p_i.timestamp = p_i.timestamp + t$
    end if
  end if
  Append p_i to T'
end for
return T'

Algorithm 1. Adding random time delay.

specifies that all dummy packets adhere to TCP protocol standards, with IP/port quintuple alignment matching the host flow. Payload content contains randomized alphanumeric strings (length: 0–20 bytes) generated through cryptographically secure methods. The formalized injection mechanism in Algorithm 2 operates under these constraints. The noise parameter $b\%$ indicates an independent per-packet insertion probability. Before transmitting each legitimate packet, the attacker independently executes dummy packet injection with probability $b\%$.

Input: A flow $T = \{p_1, \cdots, p_n\}$, the probability of noise generation $b\%$.
Output: The obfuscated flow T'
$T' = [\,]$
for $i \in N$ **do**
  $R = randit(1, 10)$
  if $R \leq 10 * b\%$ **then**
    p = new TCP packet
    S = random string from $\{a, b, .., z, A, B, ...Z\}$
    p.TCPPaylaod = S
    Appand p to T'
  end if
  Append p_i to T'
end for
return T'

Algorithm 2. Injecting dummy packets.

3.4.3 Modifying the port number

In practical situations, attackers only need to replace communication ports to achieve significant interference. Unfortunately, ports are one of the commonly used features in existing works. In Algorithm 3, for all packets in a traffic, we

uniformly modify the source port to a random value ranging from 0 to 65,535, simulating the attacker changing communication ports.

Input: A flow $T = \{p_1, \cdots, p_n\}$
Output: The obfuscated flow T'
$T' = [\,]$
$sport = randit(1, 65535)$
$dport = randit(1, 65535)$
for $i \in N$ **do**
  $p_i.sport = sport$
  $p_i.dport = dport$
  Append p_i to T'
end for
return T'

Algorithm 3. Modifying the port number.

3.4.4 Adding random payload

Although the added payload itself will not have any impact on encrypted traffic detection methods, as existing methods are not based on payload decryption, it will affect the packet length, which is a key feature. Meanwhile, several studies that extract features solely from the first N bytes of a flow face feature space degradation due to strategic payload padding. In Algorithm 4, we assume that each packet has a c% probability of adding a random payload, whose length varies between 0 and 20 bytes, with the payload generated through random sampling from a 52-character mixed-case alphabet (26 uppercase + 26 lowercase ASCII letters).

Input: A flow $T = \{p_1, \cdots, p_n\}$, the probability of noise generation c%.
Output: The obfuscated flow T'
$T' = [\,]$
for $i \in N$ **do**
  $R = randit(1, 10)$
  if $R \leq 10 * c\%$ **then**
    $payload = p_i.lastlayer.load$
    S = random string from $\{a, b, \ldots, z, A, B, \ldots, Z\}$
    $new\_payload = payload + random\_str$
    $p_i.lastlayer.load = new\_payload$
  end if
  Append p_i to T'
end for
return T'

Algorithm 4. Adding random payload.

4 Performance evaluation

In this section, we conduct experiments to evaluate the effectiveness of the proposed method. We first introduce the experimental setting and the performance metrics. Then, according to the experiment results, we analyze the confusion of four types of noise and the robustness of the proposed model.

4.1 Experiment setting and performance metrics

To validate the effectiveness of these methods, we use the well-known public dataset CICIDS2017 (Sharafaldin et al., 2018). It contains benign and the most up-to-date common attacks, which resemble real-world data. We select five types of malicious traffic with sufficient samples in the dataset, i.e., DOS, FTP-Patator, portscan, SSH-Patator, and DDOS-LOIT. We constructed the dataset by randomly selecting 2,000 samples per malicious traffic category and 8,000 benign traffic samples. The training set comprises 6,400 benign samples and 1,600 malicious samples per category, while the test set contains 1,600 benign samples and 400 malicious samples per category. All traffic payloads have undergone supplementary cryptographic processing.

We select two types of methods as comparative models, i.e., deep learning methods and traditional machine learning methods. The first type of method uses sequence features as well as the type of features we used in this study, which eliminates the need for manual feature selection. The second type of methods uses manually extracted features, e.g., statistical features.

- 1D-CNN (Zeng et al., 2019). It takes the grayscale map of traffic as features. It can learn from raw traffic without feature extraction. The CNN framework extracts critical features from raw data directly. We chose it as a comparative method to demonstrate the ability of the framework, which combines raw features with deep learning models, to resist noise.
- MalDIST (Bader et al., 2022) The system extracts three distinct feature categories: the initial 784 bytes of the traffic flow, directional patterns/sizes/inter-arrival times/TCP window sizes of the first 32 packets, and statistical characteristics derived from these 32 packets. Employing a hybrid architecture integrating CNN, GRU, and bidirectional LSTM for contextual analysis, the processed feature vectors are concatenated into a unified traffic representation that drives the final classification through dense layers with softmax activation.
- Traditional machine learning methods. Referring to commonly used statistical features in existing methods, we select features, such as average, maximum, and minimum packet length, average inter-arrival-time, average TTL, the total number of packets, port number, and the duration of traffic, as feature representation. We choose the most commonly used machine learning models, i.e., random forest and k-NN, as representatives, which combine statistical features with traditional machine learning methods.

Our experimental framework implements four adversarial traffic perturbations: (1) adding random time delay (IAT), (2) injecting dummy packets (Pkt), (3) modifying the port number (Port), and (4) adding random payload (Pld). Through adjustable noise probability parameters p, we systematically quantify feature

TABLE 1 The experiment results with different noise types and noise values.

Noise value (%)	Noise type	1D-CNN (%)	RF (%)	k-NN (%)	MalDist (%)	RD (%)
None		**89.87**	**97.77**	**97.47**	**94.43**	**97.58**
10	Port	89.86	97.19	80.72	94.18	97.58
	Pkt	49.83	72.11	62.20	83.39	84.10
	Pld	89.45	97.37	92.91	88.12	97.92
	IAT	82.93	97.78	93.52	87.18	97.57
	Avg	**78.02**	**91.11**	**82.34**	**88.22**	**94.29**
20	Port	89.86	97.19	80.72	94.18	97.58
	Pkt	49.21	71.29	61.60	81.95	83.00
	Pld	89.43	97.10	91.62	87.73	97.62
	IAT	77.43	96.63	89.56	87.07	96.46
	Avg	**76.48**	**90.55**	**80.88**	**87.73**	**93.67**
30	Port	89.86	97.19	80.72	94.18	97.58
	Pkt	89.43	96.83	90.07	87.66	97.58
	IAT	75.21	96.21	89.49	86.95	96.20
	Avg	**75.04**	**86.92**	**78.30**	**84.34**	**92.62**
50	Port	89.86	97.19	80.72	94.18	97.58
	Pkt	39.21	43.37	40.07	66.56	72.16
	Pld	89.39	96.24	89.95	87.65	97.43
	IAT	69.34	95.56	88.92	86.43	95.33
	Avg	**71.95**	**83.09**	**74.92**	**83.71**	**90.63**

The bolded parts in the table represent the average detection performance of the model for different types of obfuscated traffic.

corruption levels while simulating detection robustness across adversarial scenarios.

To evaluate the detection performance of the model, we select the F1-score and the confusion matrix as the performance metrics. F1-score is the harmonic average of precision and recall. The confusion matrix is a square matrix of (n, n), where n represents the number of classes. In the confusion matrix, columns represent the predicted classification results of the model, and rows represent the actual ground truth classification results.

4.2 Experiment results

The experiment results are shown in Table 1. We designed four types of artificial noise, and the amount of added noise is controlled through the probability. Taking the added dummy packets as an example, we modify the probability value b to ensure that there is a b% probability of each packet being inserted into the dummy packet before it. The noise value column in the table represents the probability; we set the values as 10, 20, 30, and 50. It should be noted that the interference we add to ports is not affected by probability. Once an attacker chooses this type of noise, she will replace both the source and destination ports of each packet, which is more in line with reality. Overall, from the results, we can observe that our method performs best compared with others, and the 1D-CNN model performs poorly. When there is no noise added, the RF, k-NN, and RobustDetector (RD) all achieve an F1-score of ∼97%. Among them, the F1-score of RF is slightly higher than that of RD, which may be due to interference added during the training process.

When noise is added, the F1-score of 1D-CNN decreases rapidly, while RF and RD perform relatively better. When the probability of noise increases by 50%, the F1-score of RD is 90.63%, which is ∼7% higher than RF and MalDIST. It proves that our method still has robustness even when traffic features are highly disturbed. Especially for adding dummy packets, when the probability is 30%, the F1-score of MalDIST has dropped to 68.58%, while the F1-score of RD is 79.12%, which is 11% higher than MalDIST.

From the experimental results, we can also see the interference of different types of noise on the model. Adding a random time delay, as well as modifying the port number, has minimal interference on the model. Perhaps the proportion of port numbers in traffic features is relatively small. For example, in 1D-CNN, it uses grayscale images as traffic representation, and the changes of port numbers have little effect on the grayscale images. For our method, our feature sequence did not take into account port numbers, so the changes of port numbers have no impact on RD. The F1-score of k-NN decreases by ∼17% when the port number changes, indicating that the port number is not suitable as a statistical feature for traditional machine learning methods.

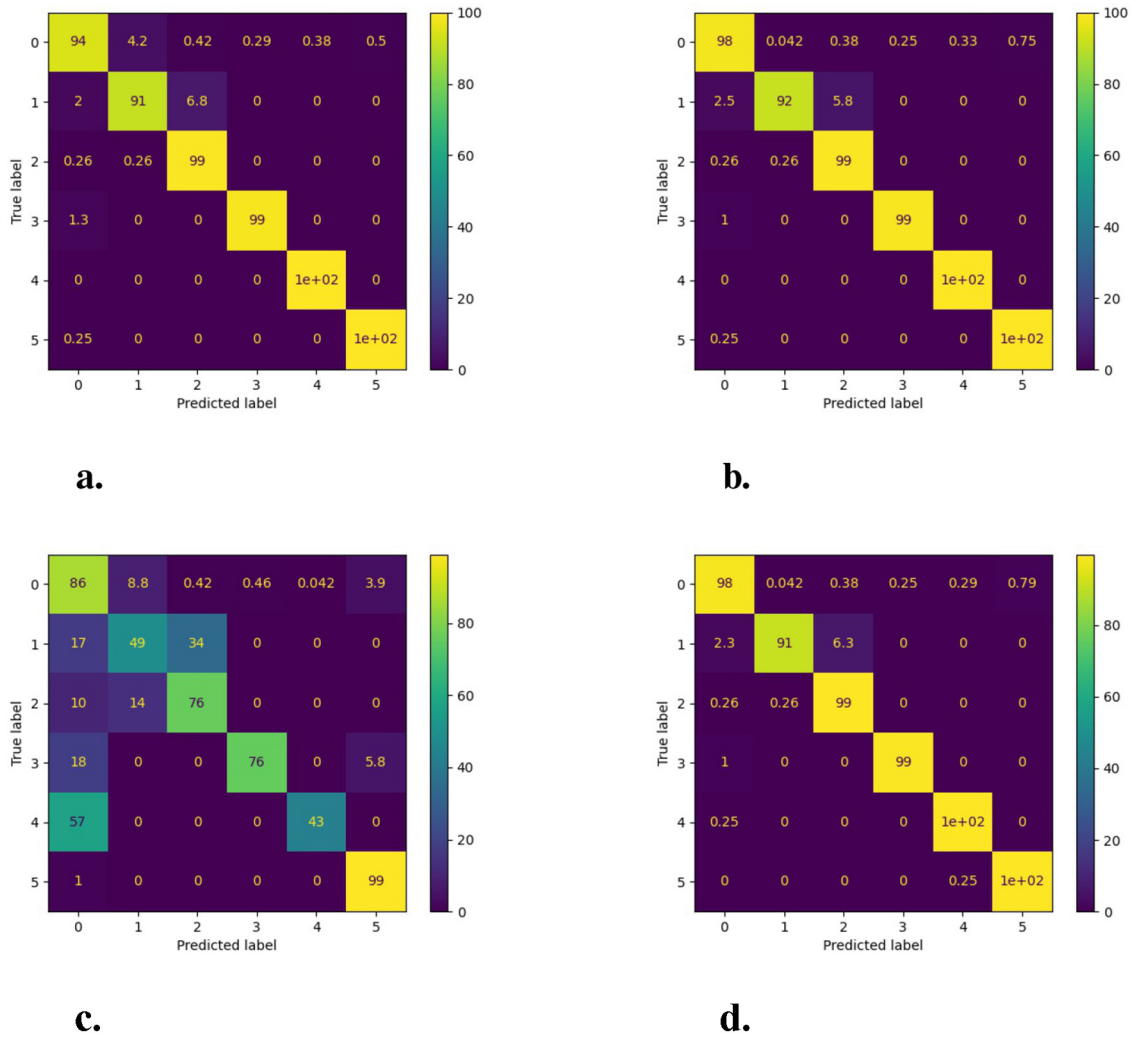

FIGURE 5
The experiment results with the noise value of 50% for RD. The experiment results for **(a)** Adding random time delay to data packets in the original traffic (IAT). **(b)** Adding random extra fields to the original packet payload (Pld). **(c)** Injecting dummy packets in the original traffic (Pkt). **(d)** Modifying the port number (Port). The probability of all noise being added is 50% refering to algorithms in Section 3.4.

The most significant impact on the model F1-score is the addition of dummy packets. When adding dummy packets, 1D-CNN is most affected, and it basically lacks classification ability as 1D-CNN leverages a grayscale image as traffic representation, which uses the first N bytes of traffic. This noise will cause the original traffic characteristics to be lost, as the added noise in the middle occupies the space of the original traffic features. Adding dummy packets also brings great challenges to RF, k-NN, and MalDIST. The addition of dummy packets will change the values of most statistical features, such as the packet length, packet number, and duration of traffic. For RD, as shown in Figure 5, adding redundant data packets also has an impact on it, but it is less affected than other methods. In this figure, we describe in detail the performance of RD under different artificial noise. The horizontal axis of each image represents the model detection result, and the vertical axis represents the true label. From 1 to 5, representing the traffic of five types of attacks in sequence, i.e., DDOS-LOIT, DOS, FTP-Patator, portscan, SSH-Patator, and label 0, represents normal traffic. Among the five categories of attack traffic, DDoS-LOIT and portscan exhibit the highest classification complexity. This challenge arises from the critical factor that both attack types generate sessions with extremely low packet volumes per flow, severely limiting detectable feature dimensions. Moreover, redundant packet injection compounds this issue, which compresses the feature space.

To demonstrate the effectiveness and robustness of the proposed method, we designed ablation experiments for comparison. Compared to the proposed method, the comparison model, named RobustDetector-W/O did not use the dropout mechanism to generate noise samples, which means that no noise is introduced during the training process. The comparison method named RobustDetector-grayscale leverages a grayscale image instead of sequence features as a traffic representation. The experiment results are shown in Table 2. When the noise value is 50, the F1-score of the RobustDetector (90.63%) is better than that of the RobustDetector-W/O (86.32%) and RobustDetector-grayscale

TABLE 2 The ablation experiment results with different noise types and noise values.

Noise value (%)	Noise type	RD-WO (%)	RD-greyscale (%)	RD (%)
None	\	**95.83**	**94.13**	**97.58**
10	Port	95.83	93.98	97.58
	Pkt	73.46	74.17	84.10
	Pld	95.89	93.86	97.92
	IAT	94.73	88.45	97.57
	Avg	**89.98**	**87.62**	**94.29**
20	Port	95.83	93.98	97.58
	Pkt	72.25	68.13	83.00
	Pld	95.87	93.82	97.62
	IAT	94.52	86.21	96.46
	Avg	**89.62**	**85.54**	**93.67**
30	Port	95.83	93.98	97.58
	Pkt	67.64	64.78	79.12
	Pld	95.74	93.73	97.58
	IAT	94.02	84.65	96.20
	Avg	**88.31**	**84.29**	**92.62**
50	Port	95.83	93.98	97.58
	Pkt	60.70	60.10	72.16
	Pld	95.71	93.65	97.43
	IAT	93.03	82.95	95.33
	Avg	**86.32**	**82.67**	**90.63**

The bolded parts in the table represent the average detection performance of the model for different types of obfuscated traffic.

(82.67%). During the training process of RobustDetector, due to the dropout mechanism playing a role in data augmentation, the number of training samples for RobustDetector is twice that of RobustDetector-W/O. This has provided RobustDetector with sufficient training. However, both models achieved good results in terms of detection results for the original samples. Compared with RobustDetector-grayscale, RobustDetector-W/O selects important features from traffic flow sequences. The grayscale images introduce a large amount of invalid information, reducing classification accuracy.

In terms of detection accuracy, RobustDetector increases the robustness of the model by introducing noise through the dropout mechanism during the training phase. By associating the original features with the obfuscated features, the ability to detect obfuscated malicious traffic has been improved. Compared with RobustDetector-W/O, the proposed method performs better when artificial noise is introduced.

5 Conclusion

In this study, we proposed RobustDetector, a novel encrypted malicious traffic detection method for obfuscated malicious traffic detection. The core of the proposed method is to use the dropout mechanism to simulate the process of original features being disturbed to improve the robustness of the model. To verify the performance of the model, we simulated four common types of artificial noise by modifying the original PCAP file, i.e., adding random time delays, injecting dummy packets, modifying the port number, and adding random payloads. From the experiment results, RobustDetector performs better in obfuscated malicious traffic classification. However, the scope of noise types considered in this study is limited. In real-world scenarios, attackers may further evade detection by employing targeted obfuscation tactics (e.g., mimicking legitimate application protocols), or injecting structured adversarial noise. Future work will expand the investigation to more complex scenarios to comprehensively validate detection robustness.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

KY: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Resources, Validation, Visualization, Writing – original draft, Writing –

review & editing. TZ: Data curation, Formal analysis, Investigation, Methodology, Writing – original draft. YD: Formal analysis, Funding acquisition, Investigation, Methodology, Writing – original draft. JH: Conceptualization, Formal analysis, Investigation, Methodology, Writing – original draft. GZ: Data curation, Formal analysis, Investigation, Methodology, Resources, Writing – original draft. ZC: Conceptualization, Formal analysis, Investigation, Methodology, Validation, Writing – original draft. YW: Investigation, Methodology, Project administration, Visualization, Writing – original draft, Writing – review & editing.

Funding

The author(s) declare that financial support was received for the research and/or publication of this article. This study was supported by the Research of Secrets Protection Project of National Administration of State Secrets Protection (Grant No. BMKY2023B07) and the Presidential Foundation of CAEP (Grant No. YZJJZQ2023026).

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declare that no Gen AI was used in the creation of this manuscript.

Publisher's note

References

Aceto, G., Ciuonzo, D., Montieri, A., and Pescapé, A. (2021). Distiller: encrypted traffic classification via multimodal multitask deep learning. *J. Netw. Comput. Appl.* 183:102985. doi: 10.1016/j.jnca.2021.102985

Ahmed, A. A., Jabbar, W. A., Sadiq, A. S., and Patel, H. (2022). Deep learning-based classification model for botnet attack detection. *J. Ambient Intell. Humaniz. Comput.* 13, 3457–3466. doi: 10.1007/s12652-020-01848-9

Anderson, B., and McGrew, D. (2017). "Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY: ACM), 1723–1732. doi: 10.1145/3097983.3098163

Arora, A., Garg, S., and Peddoju, S. K. (2014). "Malware detection using network traffic analysis in android based mobile devices," in *2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies* (Oxford: IEEE), 66–71. doi: 10.1109/NGMAST.2014.57

Bader, O., Lichy, A., Hajaj, C., Dubin, R., and Dvir, A. (2022). "Maldist: from encrypted traffic classification to malware traffic detection and classification," in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)* (Las Vegas, NV: IEEE), 527–533. doi: 10.1109/CCNC49033.2022.9700625

Chen, Z., Yan, Q., Han, H., Wang, S., Peng, L., Wang, L., et al. (2018). Machine learning based mobile malware detection using highly imbalanced network traffic. *Inf. Sci.* 433, 346–364. doi: 10.1016/j.ins.2017.04.044

Fallah, S., and Bidgoly, A. J. (2022). Android malware detection using network traffic based on sequential deep learning models. *Softw. Pract. Exp.* 52, 1987–2004. doi: 10.1002/spe.3112

Fu, Z., Liu, M., Qin, Y., Zhang, J., Zou, Y., Yin, Q., et al. (2022). "Encrypted malware traffic detection via graph-based network analysis," in *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses* (New York, NY: ACM), 495–509. doi: 10.1145/3545948.3545983

Han, S., Wu, Q., Zhang, H., and Qin, B. (2022). "Light-weight unsupervised anomaly detection for encrypted malware traffic," in *2022 7th IEEE International Conference on Data Science in Cyberspace (DSC)* (Guilin: IEEE), 206–213. doi: 10.1109/DSC55868.2022.00034

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv* [Preprint]. arXiv:1207.0580. doi: 10.48850/arXiv.1207.0580

Jung, I.-S., Song, Y.-R., Jilcha, L. A., Kim, D.-H., Im, S.-Y., Shim, S.-W., et al. (2024). Enhanced encrypted traffic analysis leveraging graph neural networks and optimized feature dimensionality reduction. *Symmetry* 16:733. doi: 10.3390/sym16060733

Lashkari, A. H., Kadir, A. F. A., Gonzalez, H., Mbah, K. F., and Ghorbani, A. A. (2017). "Towards a network-based framework for android malware detection and characterization," in *2017 15th Annual Conference on Privacy, Security and Trust (PST)* (Calgary, AB: IEEE), 233–233. doi: 10.1109/PST.2017.00035

Liu, J., Xiao, Q., Jiang, Z., Yao, Y., and Wang, Q. (2022). "Effectiveness evaluation of evasion attack on encrypted malicious traffic detection," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)* (Austin, TX: IEEE), 1158–1163. doi: 10.1109/WCNC51071.2022.9771726

Meghdouri, F., Zseby, T., and Iglesias, F. (2018). Analysis of lightweight feature vectors for attack detection in network traffic. *Appl. Sci.* 8:2196. doi: 10.3390/app8112196

Rahmat, S., Niyaz, Q., Mathur, A., Sun, W., and Javaid, A. Y. (2019). "Network traffic-based hybrid malware detection for smartphone and traditional networked systems," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)* (New York, NY: IEEE), 0322–0328. doi: 10.1109/UEMCON47517.2019.8992934

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018). "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP, Vol. 1* (Funchal), 108–116. doi: 10.5220/0006639801080116

Shen, M., Ye, K., Liu, X., Zhu, L., Kang, J., Yu, S., et al. (2022). Machine learning-powered encrypted network traffic analysis: a comprehensive survey. *IEEE Commun. Surv. Tutor.* 25, 791–824. doi: 10.1109/COMST.2022.3208196

Shi, Z., Zhao, D., Zhu, Y., Xie, G., Li, Q., and Jiang, Y. (2025). "Helios: learning and adaptation of matching rules for continual in-network malicious traffic detection," in *ACM Web Conference 2025 (WWW '25)* (New York, NY: ACM). doi: 10.1145/3696410.3714742

Xu, G., Guo, B., Su, C., Zheng, X., Liang, K., Wong, D. S., et al. (2020). Am I eclipsed? A smart detector of eclipse attacks for Ethereum. *Comput. Secur.* 88:101604. doi: 10.1016/j.cose.2019.101604

Yang, X., Peng, G., Zhang, D., Lv, Y. (2022). An enhanced intrusion detection system for iot networks based on deep learning and knowledge graph. *Secur. Commun. Netw.* 2022, 1–21. doi: 10.1155/2022/4748528

Zeng, Y., Gu, H., Wei, W., and Guo, Y. (2019). *deep — full — range*: a deep learning based network encrypted traffic classification and intrusion detection framework. *IEEE Access* 7, 45182–45190. doi: 10.1109/ACCESS.2019.2908225

Zhang, W., Meng, Y., Liu, Y., Zhang, X., Zhang, Y., Zhu, H., et al. (2018). "Homonit: Monitoring smart home apps from encrypted traffic," in *Proceedings of the 2018*

ACM SIGSAC Conference on Computer and Communications Security (New York, NY: ACM), 1074–1088. doi: 10.1145/3243734.3243820

Zhao, Z., Li, Z., Xie, X., Yu, J., Zhang, F., and Zhang, R. (2024). Foss: towards fine-grained unknown class detection against the open-set attack spectrum with variable legitimate traffic. *IEEE/ACM Trans. Netw.* 32, 3945–3960. doi: 10.1109/TNET.2024.3413789

Zhou, G., Guo, X., Liu, Z., Li, T., Li, Q., Xu, K., et al. (2025). "Trafficformer: an efficient pre-trained model for traffic data," in *IEEE Symposium on Security and Privacy (SP)* (San Francisco, CA: IEEE). doi: 10.1109/SP61157.2025.00102

Zscaler (2024). *Zscaler ThreatLabz 2023 State of Encrypted Attacks Report*. Available online at: https://info.zscaler.com/resources-industry-reports-threatlabz-2023-state-of-encrypted-attacks-report (Accessed February 28, 2024).