Training with Only 1.0 ‰ Samples: Malicious Traffic Detection via Cross-Modality Feature Fusion

Chuanpu Fu*
Department of Computer Science and Technology, Tsinghua University
Beijing, China

Qi Li
INSC and State Key Laboratory of Internet Architecture, Tsinghua University
Beijing, China

Elisa Bertino
Department of Computer Science, Purdue University
West Lafayette, Indiana, United States

Ke Xu
Department of Computer Science and Technology, Tsinghua University
Zhongguancun Lab
Beijing, China

Abstract

Machine Learning (ML) based malicious traffic detection systems can accurately recognize unseen network attacks by learning from large-scale traffic datasets. However, deploying such systems across multiple networks involves substantial efforts to construct large training datasets for each network. This paper addresses the issue of training with minimal datasets, that is, achieving accurate malicious traffic detection by learning a small portion of traffic in entirely new network environments, thereby eliminating prohibitive labor costs associated with traffic dataset construction. We develop tFusion to effectively extract information from limited datasets by treating network traffic data as multimodal data, comprising features from multiple sensory modalities of packets, flows, and hosts. In particular, we design a dedicated crossmodal attention model that fuses fine-grained per-packet sequential features with coarse-grained per-flow and per-host statistical features, to synthesize correlations among the different granularities of traffic features. Moreover, we design a topology-driven contrastive learning approach that pretrains the models while reducing topology-related biases, which allows tFusion to achieve generic detection across various networks. We deploy tFusion in an institutional network and measure its performance over five days. tFusion requires human experts to label only 1.0 ‰ traffic, yet it achieves 99.82% accuracy when detecting various attacks. Meanwhile, it outperforms 14 existing methods by improving over 12.76% accuracy on 11 existing datasets.

CCS Concepts

• **Security and privacy → Network security**.

Keywords

Network security; machine learning; malicious traffic detection

ACM Reference Format:

Chuanpu Fu, Qi Li, Elisa Bertino, and Ke Xu. 2025. Training with Only 1.0 ‰ Samples: Malicious Traffic Detection via Cross-Modality Feature Fusion . In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), October 13–17, 2025, Taipei, Taiwan.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3719027.3765143

*Chuanpu Fu completed this research while serving as a visiting scholar at Purdue University.

CCS '25, Taipei, Taiwan

ACM ISBN 979-8-4007-1525-9/2025/10
https://doi.org/10.1145/3719027.3765143

1 Introduction

Machine learning (ML) based malicious traffic detection is an important security technique that inspects traffic generated by various protocols and applications to recognize malicious flows with abnormal traffic features [8, 38, 79, 98, 107]. Over more than a decade of development, systems based on this technique have been deployed at network gateways to protect critical services [33, 34, 36], where they have successfully throttled real-world attack traffic [1, 2, 16] and outperformed traditional rule-based detection [88]. Currently, the market for malicious traffic detection is valued at more than $3.64 billion, with expectations for expanding deployment across various networks [66].

Unfortunately, expanding the deployment of malicious traffic detection systems to new networks incurs prohibitive labor costs. Specifically, human experts must capture and label millions of benign and attack packets [43, 45, 107], because large-scale traffic datasets are essential for ML models to learn the complex traffic patterns generated by various applications and protocols in new network environments. This labor-intensive process [5, 10, 70] impedes a scalable deployment to protect vast numbers of Internet users from malicious traffic [5, 78].

To address this scalability problem, it is necessary to minimize the scale of the required training traffic datasets. *Therefore, the goal of our work is to develop a detection system that maintains high precision, even when trained on a very small amount of traffic samples from the network environment of interest.*

Our solution is based on the notion of multimodal AI, that is, a cognitive process that synthesizes patterns from multiple sensory modalities, such as texts, sounds, and images. We observe that existing detection methods still treat traffic as unimodal data, by exclusively learning per-packet [43, 100], per-flow [8, 107], or per-host feature vectors [33, 34] (see Table 1). Consequently, the inability to synthesize the different granularities of features results in significant information loss, requiring large-scale training datasets [35, 64, 107]. In contrast, we view bytes delivered in networks from multiple modalities by extracting heterogeneous traffic features from the perspectives of packets, flows, and hosts. Subsequently, we correlate the traffic features across the different modalities to enrich the information available from limited datasets.

This paper presents tFusion, a malicious traffic detection system that effectively extracts information from limited training datasets.

Table 1: The comparison with the state-of-the-art ML based malicious traffic detection methods.

Categories	Methods	ML Models	Dataset Requirements			Detection Abilities				Performances	
			Feature Modality	Feature Scale	w/o Large Datasets	Generic Detection	Evasion Attack	Unknown Attack	Encrypted Traffic	Realtime Detection	Efficient Detection
Supervised Learning	nPrintML [43]	AutoML	Packet	$O(10^5)$	×	✓	×	×	✓	✓	×
	FlowLens [8]	Random Forest	Flow	$O(10^4)$	×	✓	×	×	✓	×	✓
	Taurus [79]	SVM	Flow	$O(10^4)$	×	✓	×	×	×	✓	✓
	NetBeacon [107]	Decision Tree	Flow	$O(10^7)$	×	✓	×	×	×	✓	✓
	BoS [100]	RNN	Packet	$O(10^4)$	×	✓	×	×	✓	✓	✓
	N3IC [77]	Binary DNN	Flow	$O(10^6)$	×	✓	×	×	×	✓	✓
Unsupervised Learning	Whisper [33]	K-Means	Host	$O(10^3)$	×	✓	✓	✓	×	✓	✓
	Kitsune [64]	AutoEncoder	Packet	$O(10^5)$	×	✓	×	✓	×	×	×
	EULER [53]	GNN+RNN	Host	$O(10^3)$	×	×	×	✓	✓	✓	×
	HorusEye [22]	Isolation Forest	Flow	$O(10^5)$	×	×	×	✓	×	✓	✓
	HyperVision [34]	Graph Learning	Host	$O(10^7)$	×	✓	×	✓	✓	×	✓
Both	**tFusion** (Ours)	Crossmodal Model	ALL	$O(10)$	✓	✓	✓	✓	✓	✓	✓

In particular, tFusion fuses traffic features with different granularities by using a pre-trained crossmodal model [18, 87, 92]. Specifically, the crossmodal model employs the attention mechanism to extract correlations among the different granularities of features. More precisely, the model calculates weights (attentions) for fine-grained packet features based on coarse-grained flow and host characteristics, to highlight critical parts of packet sequences. In this way, tFusion synthesizes correlations among multiple modalities to obtain rich information from limited datasets, thus providing effective features to train both supervised and unsupervised lightweight ML detection models to identify various malicious behaviors.

However, correlation analysis for multimodal data, which comprises sequential and non-sequential traffic features, is a non-trivial task. First, to extract sequential features, we develop a time-aware positional encoding to synthesize packet-level temporal and spatial patterns. Second, to fuse the sequential and non-sequential features, we design a crossmodal attention model that employs host features as queries and flow features as keys to compute attentions, which allows tFusion to focus on critical regions of packet sequences associated with various flows. Third, to pre-train the attention model, we design a topology-driven contrastive learning approach that utilizes large-scale unlabeled Internet traffic datasets. This minimizes the need of labeling traffic for many networks. Specifically, our topology-driven pretraining guides the model to cluster traffic sent by same addresses in the feature space, regardless of ground-truth labels. Finally, during the deployment phase, we leverage Automatic ML (AutoML) [43] to select models for learning the features extracted by the pre-trained model from minimal datasets.

To assess the performance of tFusion, we replay 11 existing datasets to compare it with 14 state-of-the-art methods. The experimental results show that tFusion outperforms existing methods by improving the accuracy of 12. 76% in various training dataset settings. Even in critical settings, such as learning from 50 samples and 1.0 ‱ traffic, tFusion still retains more than 94. 92% precision and robustness against evasion attacks [33]. Moreover, tFusion significantly outperforms data augmentation methods [6, 46, 95] that mitigate the data scarcity issue for other ML tasks. Furthermore, the overall detection latency of tFusion is 34ms, which is comparable to the latency of existing detection systems [33, 34]. Finally, we deploy tFusion on an institutional network with more than 140 active users and measure its performance over five days. tFusion requires an expert team to label 1.0 ‰ flows (91 captured flows) for training an unsupervised model. Once trained, the model achieves 99.82% accuracy when detecting 25 manually constructed attacks.

In summary, the contributions of this paper are five-fold:

- We utilize crossmodal attention to correlate traffic features across different granularities, enabling malicious traffic detection with limited training datasets.
- We develop a time-aware packet sequence embedding to extract fine-grained packet features.
- We design a cross-attention to fuse the packet features and coarse-grained statistical features for correlation analysis.
- We pre-train the attention model using large-scale unlabeled Internet traffic via topology-driven contrastive learning.
- We prototype[1] tFusion and validate its accuracy and efficiency in various network environments.

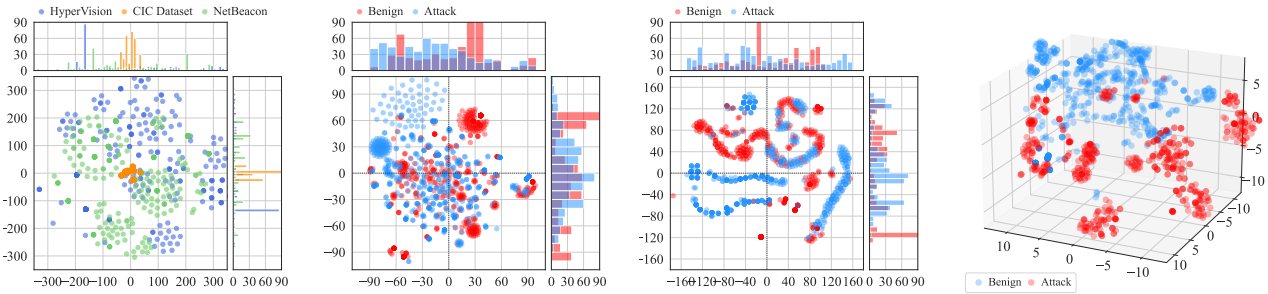
Additionally, our key idea, that is, seeing traffic as multimodal data, may also reduce the dependency of large datasets for other learning tasks, e.g., classifying benign application traffic [104]. We present initial comparative experiments to suggest future research.

Note that tFusion is not intended to outperform existing systems by simply devising similar flow or packet features. Instead, we aim to eliminate the dependency on large-scale training datasets by correlating different granularities of traffic features, thereby promoting broad deployment across various networks.

Road Map. The rest of the paper is organized as follows. Section 2 introduces the threat model, the goals of our design, and the related solutions. Sections 3 and 4 present an overview and the detailed design of tFusion, respectively. Section 5 presents the results of our experimental evaluation of tFusion. Section 6 discusses limitations. Section 7 reviews related works and Section 8 outlines conclusions.

Ethical Issues. We generate attack traffic within a separate subnet, and directly forward the traffic to the machine where both malicious and benign traffic is collected. In addition, firewalls are configured to prevent interference with real users (see Section 5.1). In addition, we do not disclose personally identifiable information (PII) of users and experts. The users and administrators consented to our experiments and the release of the dataset. Furthermore, our system identified

[1]Source code and datasets: https://github.com/fuchuanpu/TFusion.



(a) Traffic from different datasets. (b) Packet feature space. (c) Flow feature space. (d) Joint feature space.

Figure 1: TSNE visualized feature spaces: packet [100] and flow features [28] are extracted from the dataset [34] by default.

unusual traffic which may relate to network events (e.g., routing updates). We reported these events to the administrators.

2 Problem Statement

Threat Model. We assume that attackers generate various malicious network traffic flows from compromised machines [11, 39, 81]. Note that the attackers may control varying scales of machines, resulting in highly variable traffic features [49]. In addition, we assume that the attacker is able to generate stealthy traffic using sophisticated strategies, such as generating encrypted traffic [34, 81], exploiting advanced vulnerabilities [13, 26], crafting behaviors that mimic benign users [51, 60], and applying adversarial strategies for evasion, e.g., manipulating features with traffic obfuscation [33] and concealing topology patterns using tunnels [37].

Design Goals. The design of tFusion should meet several requirements: (1) It should achieve accurate detection by learning from limited training datasets. (2) It should support unsupervised ML, i.e., learning from samples with benign labels in the absence of attack samples [40, 64]. (3) It should meet the traditional requirements for malicious traffic detection that are listed in Table 1: (i) Generic detection for various attacks with different speeds and durations; (ii) Robust detection for existing evasion attacks [33, 35]; (iii) Accurate detection for zero-day attacks [80]; (iv) Ability to capture both encrypted and plain-text malicious traffic [34]; and (v) Low-latency detection in high-speed networks.

In general, tFusion focuses on detecting unknown attack traffic flows among high-speed Internet traffic flows generated by various protocols and applications in real time. This task is fundamentally different from off-line supervised traffic classifications for specific known kinds of applications [58, 75, 104]. Moreover, unlike the classification of benign applications [58, 105], tFusion should not directly learn raw bytes in packet payloads [42, 108], as attackers can easily manipulate data in the form of raw bytes for evasion attacks [5, 44].

Potential Solutions. We observe that complementing limited datasets with samples from other datasets cannot mitigate the problem of data scarcity because the distributions of traffic features in computer networks differ significantly from each other due to the diversity of network services (see the flow features [28] in Figure 1(a)). Moreover, existing data augmentation methods [6, 48, 95] can only generate a particular type of traffic based on prior knowledge, such as Tor traffic [6] and TLS traffic [95]. Thus, these methods cannot be applied to traffic detection systems, which must handle various types of traffic for generic detection (see the experiments in Section 5.2). Unlike these methods, we seek to effectively extract informative traffic features from limited datasets.

3 Overview

3.1 Key Observations

Existing approaches view traffic as unimodal data [90], exclusively analyzing homogeneous tabular data with a single granularity, such as per-packet features [64] and per-flow features [28]. However, as we can see from Figures 1(b) and 1(c), these features are densely distributed and thus cannot effectively distinguish benign and malicious traffic. Therefore, large amounts of samples are required by ML models to regress the complex decision boundaries.

In contrast, we treat traffic as miltimodal data. That is, we view bytes delivered by networks from multiple modalities, i.e., from the perspective of packets, flows, and hosts, to extract heterogeneous features comprising different granularities of sequential and non-sequential traffic features. From Figure 1(d), we observe that the distribution of the multimodal features is more sparse. This suggests that analyzing features across multiple modalities may compensate for the information loss that occurs with traditional unimodal feature analysis, which is a promising direction for effective training with limited datasets.

3.2 High-Level Architecture

We develop tFusion to synthesize traffic features extracted from multiple perspectives. Specifically, we design a crossmodal model to capture the correlations among sequential and non-sequential features based on the attention mechanism[2].

Like existing systems [22, 33, 107], tFusion is deployed at Internet gateways, where it inspects traffic mirrored by routers [15], such as those connecting ASes through optical fibers [94]. When tFusion identifies abnormal traffic, it cooperates with existing defense systems [59, 97, 103] to effectively throttle the traffic. tFusion has four modules, as illustrated in Figure 2.

Time-Aware Packet Sequence Embedding. tFusion starts by extracting fine-grained sequential features. Specifically, we design a time-aware positional encoding algorithm to simultaneously embed both temporal information (i.e., packet-arrival intervals) and spatial information (i.e., packet sizes). Meanwhile, this module extracts packet-level sequential information by analyzing the dependencies among the packets in a flow, which facilitates the detection of stealthy attacks.

[2] Appendix A elaborates detailed procedures of attention based correlation analysis.

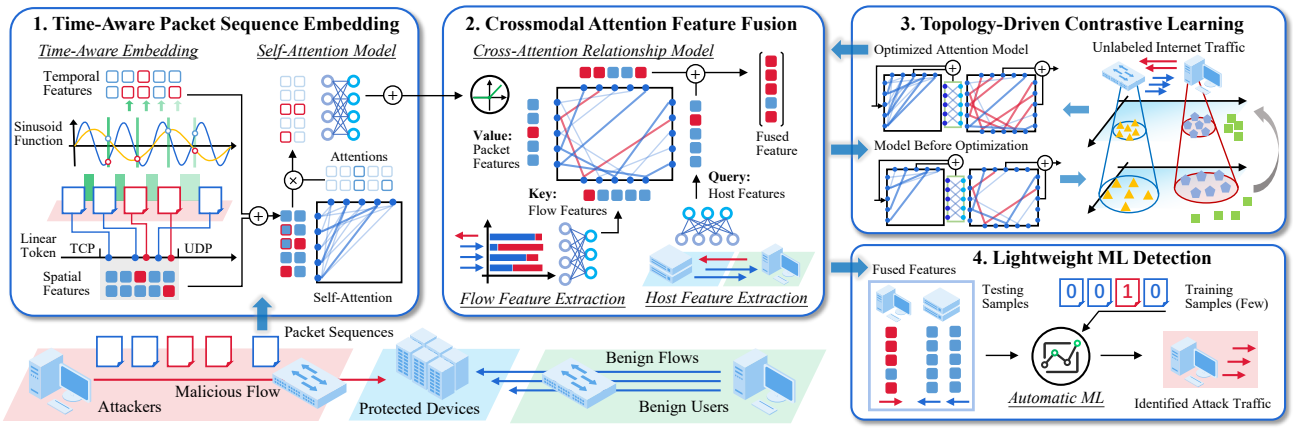


Figure 2: High-level architecture of tFusion malicious traffic detection system.

Crossmodal Attention Feature Fusion. In this module, we analyze the correlations between the fine-grained sequential features and coarse-grained non-sequential features. Specifically, we design a crossmodal attention mechanism to fuse these features with different granularities into a unified feature vector. More precisely, we use host features as queries, flow features as keys, and packet features as values. In this way, host features instruct the model to concentrate on important packets in various flows, allowing the fused feature vector to effectively represent multimodal traffic data.

Topology-Driven Contrastive Learning. To pre-train the model, we develop a contrastive learning approach, which enables utilizing large-scale unlabeled Internet traffic datasets generated by numerous benign and malicious sources. Note that the unlabeled traffic is collected from networks other than where tFusion will be deployed, thereby minimizing the labor costs of labeling training samples for many different networks. Specifically, our approach is driven by specific network topological information, that is, the traffic generation address. It gradually guides the model to gather traffic generated by same addresses in the feature space, regardless of their labels. In this way, tFusion can utilize the traffic generated by various topologies in large data sets to eliminate bias on network topologies for accurate detection in different networks.

Lightweight ML Detection. In the training phase, human experts randomly sample a small minority of traffic from new deployment networks. After labeling the traffic samples, they apply the pre-trained model to extract traffic features from the constructed dataset. The extracted features are subsequently input into an AutoML framework, which trains either four supervised ML models [8, 36, 79, 107] or three unsupervised models [22, 33, 34], depending on the availability of malicious traffic samples. These lightweight models are widely used by existing traffic detection approaches to realize efficient detection. Ultimately, the framework selects the model with the highest accuracy, which learns the features containing rich information extracted from the limited datasets.

4 Design Details

4.1 Time-Aware Packet Sequence Embedding

This module extracts fine-grained features by synthesizing spatial and temporal patterns from packet sequences. First, we aggregate packets into flows. The flows consist of packets that share the same five-tuple attributes: transport layer protocol identifier, source and destination IP addresses, and associated port numbers. The i^{th} arrived flow is denoted by set $\mathcal{F}_i = \{\mathcal{I}_i, \mathbf{P}^{(i)}\}$, where $\mathcal{I}_i = \langle \mathrm{p}, \mathrm{s}, \mathrm{d}, \mathrm{s_p}, \mathrm{d_p} \rangle$ is the tuple that characterizes the flow and the matrix $\mathbf{P}^{(i)} = [\ \vec{t}^i, \vec{l}^i\ ]$ contains the per-packets features, i.e., $\vec{t}^i$ and $\vec{l}^i$ denote the arrival timestamps and the lengths associated with the N_i packets in the i^{th} flow, respectively. Note that, we do not directly use packet header fields as features, such as port numbers and time-to-live (TTL), because analyzing these fields incurs overfitting issues [5, 44]. Instead, we extract sequential features from matrix $\mathbf{P}^{(i)}$.

Time-Aware Packet Embedding. We convert the features of each packet into a vector. This vector contains both the packet length information and the arrival time information. First, to effectively embed the length information, we offset the length based on the protocol identifier before embedding:

$$\vec{v}^i_j = \mathcal{I}_i.\mathrm{p} \times \mathrm{M} + \min(\vec{l}^i_j, \mathrm{M}) - 1, \tag{1}$$

where M denotes the maximum length of a packet that can be delivered by the network. Next, we apply a traditional embedding layer that maps each integer in $\vec{v}^i$ to a floating-point vector, forming a matrix: $\mathbf{E}^{(i)} = \mathrm{Embed}(\vec{v})$. Note that the layer is trained to maximize the distances among the mapped vectors to effectively represent the packets.

Second, we embed the temporal information, because $\mathbf{E}^{(i)}$ cannot indicate the order of packets and time intervals among the packets. Traditional positional encoding for language [18] only considers the order, but cannot consider time-scale information. Therefore, we design a time-aware positional encoding algorithm, which converts timestamps into discrete values that not only indicate a packet's position in a flow but also highlights significant time intervals between packets:

$$\vec{u}^i_j = \begin{cases} 0, & \text{if} \quad j = 1, \\ 2 \cdot j + 1, & \text{elseif} \quad T \le \vec{t}^i_j - \vec{t}^i_{j-1}, \\ 2 \cdot j, & \text{else}, \end{cases} \tag{2}$$

where $1 \le j \le N_i$. Afterward, we map each element in $\vec{u}^i$ to a vector using a sinusoidal function, which allows two packets with non-adjacent positions or large arrival intervals to be represented

by significantly different vectors:

$$\mathbf{U}_{j,p} = \begin{cases} \sin\left(p \cdot e^{-\frac{j \cdot \ln T}{H}}\right), & \text{if} \quad j \bmod 2 = 0, \\ \cos\left(p \cdot e^{-\frac{j \cdot \ln T}{H}}\right), & \text{else}, \end{cases} \tag{3}$$

where $1 \le p \le H$ and $1 \le j \le N_i$. In this way, the embedding function can effectively represent the index and time of each packet, since it utilizes exponential functions to amplify the values, and sinusoidal functions for normalization.

Finally, we compute the time-position embedding matrix as $\mathbf{T}^{(i)} = \mathbf{U}_{\vec{u}^i}$, and combine it with the length embedding matrix $\mathbf{E}^{(i)}$ to produce the matrix representing the packet sequence: $\mathbf{S}^{(i)} = \mathbf{T}^{(i)} + \mathbf{E}^{(i)}$. Note that $\mathbf{S}^{(i)}_j$ is the vector representing j^{th} packet in i^{th} flow.

Sequential Feature Extraction. For each flow, we extract sequential features to represent the relationships among the packets. We employ a self-attention model [87]. First, the model converts an input matrix into query, key, and value matrices. Afterward, it computes an attention matrix that indicates correlations, by multiplying the three matrices with non-linear transformations. Given that the scale of our embedding vectors is smaller than those used in language-related tasks [18], we utilize a smaller model to mitigate overfitting issues [5].

First, we convert matrix $\mathbf{S}^{(i)}$ into query, key, and value matrices, i.e., the input of the attention model:

$$\mathbf{Q}^{(i)} = \mathsf{Linear}\left(\mathbf{S}^{(i)}; \mathbf{W}^{(Q)}, \vec{b}^{(Q)}\right) = \mathbf{S}^{(i)}\mathbf{W}^{(Q)\mathsf{T}} + \vec{b}^{(Q)}, \tag{4}$$

where $\mathbf{W}^{(Q)}_{H\times H}$ and $\vec{b}^{(Q)}$ are trainable parameters, denoting the weights and bias of the fully connected layer with H states.

$$\mathbf{K}^{(i)} = \mathsf{Linear}\left(\mathbf{S}^{(i)}; \mathbf{W}^{(K)}, \vec{b}^{(K)}\right), \ \mathbf{V}^{(i)} = \mathsf{Linear}\left(\mathbf{S}^{(i)}; \mathbf{W}^{(V)}, \vec{b}^{(V)}\right). \tag{5}$$

In the next step, we calculate the self-attention values according to the key, value, and query matrices:

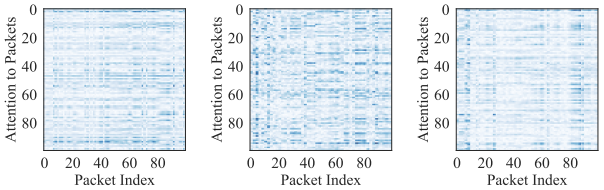
$$\mathbf{A}^{(i)} = \mathsf{Atten}(\mathbf{Q}^{(i)}, \mathbf{K}^{(i)}, \mathbf{V}^{(i)}) = \mathsf{Softmax}\left(\frac{\mathbf{Q}^{(i)}\mathbf{K}^{(i)\mathsf{T}}}{\sqrt{H}}\right)\mathbf{V}^{(i)}, \tag{6}$$

where $\mathbf{A}^{(i)}$ denotes the values of attentions. Afterward, we combine the attention matrix $\mathbf{A}^{(i)}$ with the embedding matrix $\mathbf{S}^{(i)}$. Afterward, we apply the residual connection that directly adds the model's input to its output, a widely used technique for improving deep learning models [18, 87]. Moreover, we employ a fully connected layer with $4H$ hidden states to produce the output of this module:

$$\begin{aligned} \mathbf{M}^{(i)} &= \mathsf{GELU}\left(\mathsf{Linear}\left(\mathbf{A}^{(i)} + \mathbf{S}^{(i)}; \mathbf{W}^{(1)}_{(H\times 4H)}, \vec{b}^{(1)}\right)\right), \\ \mathbf{U}^{(i)} &= \mathsf{Softmax}\left(\mathsf{Linear}\left(\mathbf{M}^{(i)}; \mathbf{W}^{(2)}_{(4H\times H)}, \vec{b}^{(2)}\right) + \mathbf{A}^{(i)}\right). \end{aligned} \tag{7}$$

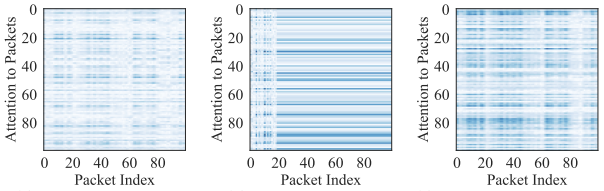
Note that $\mathsf{GELU}(\cdot)$ denotes the Gaussian Error Linear Unit function. Finally, we select the first dimension of the matrix $\mathbf{U}^{(i)}$, which is denoted by $\vec{p}_i$, to represent the fine-grained sequential information extracted from the flow.

We plot the attentions in Figure 3 and Figure 4, where a darker pixel indicates a higher attention value. We observe that a packet does not exhibit significant correlation to itself, as the values on the diagonals of the attention matrices are not significantly higher.



(a) HTTP download. (b) HTTPS web access. (c) UDP video flow.

Figure 3: Attention values associated with benign flows.



(a) Pulsing TCP DoS. (b) Telnet injection. (c) Amplification attack.

Figure 4: Attention values associated with attack flows.

Moreover, we can see the regular and periodic patterns in the attentions associated with malicious flows, such as the periodic bursts generated by pulsing attacks [60] (see Figure 4(a)). Meanwhile, the attention values of malicious flows are generally higher than those of benign flows, because attention models can easily capture their regular patterns. Note that tFusion captures both temporal and spatial packet patterns, which are critical for detecting advanced malicious traffic [8, 33]. In contrast, existing attention based benign application classification methods [17, 58, 104, 105] can only analyze the spatial patterns.

4.2 Crossmodal Attention Feature Fusion

In this module, we extract coarse-grained statistics from the perspectives of hosts and flows. Subsequently, we fuse the sequential features with these non-sequential features, by designing a cross-modal attention model.

Flow Statistical Feature Extraction. For each flow $\mathcal{F}_i$, we extract flow-level statistics from the per-packet feature matrix $\mathbf{P}^{(i)}$. Specifically, we calculate six flow features [28, 34]. The flow-level feature vector $\vec{f}$ can be denoted as:

$$\vec{f}^i = \mathsf{Log}\left(\left[N_i, \vec{t}^i_{N_i} - \vec{t}^i_1, \sum_{j=1}^{N_i}\vec{l}^i_j, \min(\vec{l}^i), \max(\vec{l}^i), \frac{1}{N_i}\sum_{j=1}^{N_i}\vec{l}^i_j\right]^{\mathsf{T}} + \vec{1}\right). \tag{8}$$

The features are: (i) the number of packets in a flow (i.e., N_i); (ii) flow completion time (FCT) denoted by $\vec{t}^i_{N_i} - \vec{t}^i_1$; (iii) the number of bytes in a flow: $\mathsf{Sum}(\vec{l}^i) = \sum_{j=1}^{N_i}\vec{l}^i_j$; the (iv) maximum, (v) minimum, and (vi) average packet lengths in the flow, denoted by $\min(\vec{l}^i)$, $\max(\vec{l}^i)$, and $\mathsf{mean}(\vec{l}^i)$, respectively. In addition, we perform logarithmic transformations to improve numerical stability.

Host Feature Extraction. We extract host-level statistics to represent flow interactions among numerous hosts. Therefore, for the flows that arrive within a small time window L, we calculate 12 host-level flow interaction features. Specifically, we define the functions $\mathsf{f}_{\mathsf{src}}(s;\mathscr{F})$ and $\mathsf{f}_{\mathsf{dst}}(d;\mathscr{F})$, which query the flows with the same source and destination hosts among all the flows in $\mathscr{F}$:

$$\begin{aligned} \mathsf{f}_{\mathsf{src}}(s;\mathscr{F}) &= \{\mathcal{F}_i | \forall \mathcal{F}_i \in \mathscr{F}, \mathcal{I}_i.s = s\}, \\ \mathsf{f}_{\mathsf{dst}}(d;\mathscr{F}) &= \{\mathcal{F}_i | \forall \mathcal{F}_i \in \mathscr{F}, \mathcal{I}_i.d = d\}. \end{aligned} \tag{9}$$

Based on the flows with the same source and destination, we define the functions that extract the features of sending and receiving patterns for one particular host:

$$\text{send}(h) = \left[|\mathsf{f}_{\text{src}}(h;\mathscr{F})|, \sum_{\mathcal{F}_j \in \mathsf{f}_{\text{src}}(h;\mathscr{F})} N_j, \sum_{\mathcal{F}_j \in \mathsf{f}_{\text{src}}(h;\mathscr{F})} \sum_{k=1}^{N_j} \vec{l}_k^j \right]^{\mathsf{T}},$$
$$\text{receive}(h) = \left[|\mathsf{f}_{\text{dst}}(h;\mathscr{F})|, \sum_{\mathcal{F}_j \in \mathsf{f}_{\text{dst}}(h;\mathscr{F})} N_j, \sum_{\mathcal{F}_j \in \mathsf{f}_{\text{dst}}(h;\mathscr{F})} \sum_{k=1}^{N_j} \vec{l}_k^j \right]^{\mathsf{T}}. \tag{10}$$

Afterward, we define the host-level feature associated with one flow $\mathcal{F}_i$ as the sending and receiving features associated with its source $\mathcal{I}_i.s$ and destination $\mathcal{I}_i.d$:

$$\vec{h}^i = \text{Log}\left(\text{Cat}(\text{send}(\mathcal{I}_i.s), \text{receive}(\mathcal{I}_i.s), \text{send}(\mathcal{I}_i.d), \text{receive}(\mathcal{I}_i.d))\right), \tag{11}$$

where the function $\text{Cat}(\cdot)$ concatenates all these vectors. Similar to extracting flow features, we also perform the logarithmic transformations for the host features.

Cross-Modality Feature Fusion. In the last step, we fuse the non-sequential features $\vec{f}^i$ and $\vec{h}^i$ with the sequential features $\vec{p}^i$, to capture the relationships among these features at different granularities. Specifically, we devise a crossmodal attention model to correlate different granularities of features by using the host features as queries, flow features as keys, and packet-level sequential features as values. This is because the flow and host features are coarse-grained, and thus cannot directly indicate the fine-grained patterns. However, these features can guide the model to assign high attention values (weights) to focus on critical packets. Note that unlike the self-attention mechanism employed to extract sequential features, the key, query, and value matrices in the cross-attention feature fusion model originate from distinct modalities, enabling effective fusion for heterogeneous data:

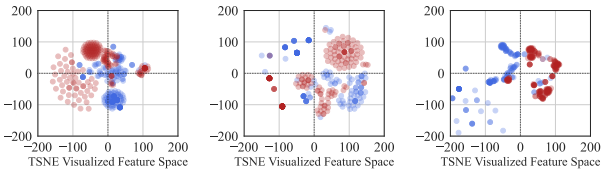
$$\vec{q}^i = \text{Linear}(\vec{h}^i; \mathbf{W}^q_{12\times H}, \vec{b}^q) = \vec{h}^{i\mathsf{T}}\mathbf{W}^q_{12\times H} + \vec{b}^q,$$
$$\vec{k}^i = \text{Linear}(\vec{f}^i; \mathbf{W}^k_{6\times H}, \vec{b}^k), \quad \vec{v}^i = \text{Linear}(\vec{p}^i; \mathbf{W}^v_{H\times H}, \vec{b}^v). \tag{12}$$

Here, we use the Hadamard product to substitute the matrix multiplication in the original attention designed for sequential data [18, 87], to support applying attentions for fusing features of heterogeneous traffic data:

$$\vec{C}^i = \text{CrossAtten}(\vec{q}^i, \vec{k}^i, \vec{v}^i) = \text{Softmax}\left(\frac{\vec{q}^i \circ \vec{k}^i}{\sqrt{H}}\right), \ \vec{F}^i = \vec{C}^i \circ \vec{v}^i + \vec{q}^i. \tag{13}$$

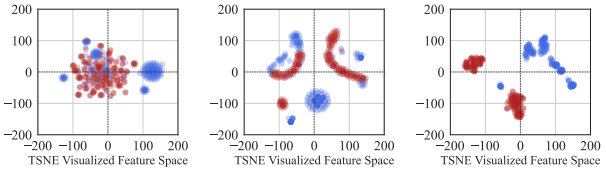
Inspired by large language models [18], we apply residual connections to enhance the relationships between the queries and attention values. Finally, $\vec{F}^i$ denotes the fused traffic features.

In Figure 5, we plot the distribution of traditional sequential [100] and non-sequential [28] features, as well as tFusion features. Specifically, we randomly select 200 benign and malicious flows and project the high-dimensional traffic features spaces. From Figure 5(a) and Figure 5(b), we can see that the feature vectors are distributed in small regions, because the crossfire attack [51] mimics benign traffic features, i.e., it congests critical links by instructing many compromised devices to simultaneously generate massive normal flow. Similarly, the amplification attack triggers massive flooding traffic by sending few packets at a low speed [72], making the associated features closer to benign ones (see Figures 6(a) and 6(b)).



(a) Sequential features. (b) Statistical features. (c) tFusion features.

Figure 5: Crossfire attack flows in the traffic feature spaces.



(a) Sequential features. (b) Statistical features. (c) tFusion features.

Figure 6: DNS amplification attack flows in the spaces.

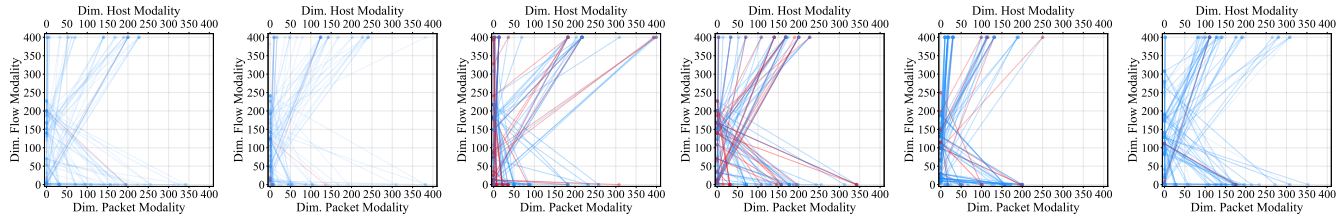
Consequently, existing approaches require large-scale datasets to learn the complex decision boundaries. In contrast, as we can see from Figures 5(c) and 6(c), the attack traffic deviates the normal relationships among the features, and thus tFusion can effectively capture the attacks by synthesizing these features.

Furthermore, we visualize the cross-attention to investigate modality-wise correlation. Initially, we extract features of benign traffic from Internet datasets [94] and our institutional network testbed (see Section 5.1), as well as malicious traffic from existing datasets [34]. Afterward, we transform the features according to Eq. 12 to calculate attention. In Figure 7, each line connecting the three modality dimensions denotes the attention among packet, flow, and host features, where transparency of the lines indicates values of attentions, and red color highlights significantly high attention values (close to 1.0 after normalization). Comparing benign attentions (Figures 7(a) and 7(b)) to abnormal ones (Figures 7(c) ~ 7(f)), we can see that values of cross-attention associated with malicious traffic are generally higher than those of benign traffic. These high attention values indicate abnormal correlations among different granularities of traffic features, and thus can be accurately captured by tFusion.

Note that existing approaches, which either correlate unimodal raw bytes within same flows [58, 105] or concatenate unimodal packet-level features of bytes and sizes [41], differ significantly from our cross-modality model which captures correlations among multiple modalities of packets, flows, and hosts. Additionally, our ablation studies show that simply concatenating features [77] fails to synthesize the correlations among modalities, and thus cannot achieve effective training on minimal datasets.

4.3 Topology-Driven Contrastive Learning

We design a contrastive learning approach that allows us to utilize large-scale unlabeled Internet traffic datasets to pre-train the attention models. This ensures that the models provide informative features for training AutoML module with minimal datasets. Unlike existing pre-training approaches [17, 48], we do not aim to achieve high accuracy on the detection task for a particular network, which restrains the accuracy in other networks. Instead, our topology-driven method pre-trains the models to maximize the distances between the features associated with flows generated by different



(a) Internet benign traffic. (b) Testbed benign traffic. (c) Crossfire attack. (d) Amplification attack. (e) Password cracking. (f) Spam traffic.

Figure 7: Attention among different modalities synthesized by the fusion model, i.e., packet, flow, and host features.

addresses, while minimizing the distances between flows generated by same addresses in the feature space.

Contrastive Sample Batch Construction. First, we aggregate samples for pre-training into batches of size $2B$. Each batch contains an identical amount of flows sent by (or received by) same and different addresses. We use $\mathscr{F}_T = \{\mathcal{F}_1, \ldots, \mathcal{F}_Y\}$ to denote all the flows in an unlabeled large-scale Internet traffic dataset, where each flow may either be benign or malicious. We first identify the source and destination addresses that send or receive more than B flows:

$$\begin{aligned} \mathcal{S} &= \{\mathcal{F}_i.s \mid |\mathsf{f}_{\mathsf{src}}(\mathcal{F}_i.s; \mathscr{F}_T)| \geq B,\ \forall \mathcal{F}_i \in \mathscr{F}_T\}, \\ \mathcal{D} &= \{\mathcal{F}_i.d \mid |\mathsf{f}_{\mathsf{dst}}(\mathcal{F}_i.d; \mathscr{F}_T)| \geq B,\ \forall \mathcal{F}_i \in \mathscr{F}_T\}. \end{aligned} \tag{14}$$

Note that B is the batch size hyperparameter. In the next step, we evenly sample flows in $\mathcal{S}$ to construct $\mathscr{S}$ containing flows sent by and not sent by same addresses:

$$\mathscr{S} = \bigcup_{\forall s \in \mathcal{S}} \{\mathscr{S}_s\}, \quad \mathscr{S}_s = \{\mathscr{S}^s_{\mathsf{same}}, \mathscr{S}^s_{\mathsf{diff}}\}, \tag{15}$$

where $\mathscr{S}_s$ denotes sampled flows associated with $s \in \mathcal{S}$. It contains $\mathscr{S}^s_{\mathsf{same}}$ (the B flows sent by s) and $\mathscr{S}^s_{\mathsf{diff}}$ (the B flows not sent by s):

$$\mathscr{S}_s = \{\mathsf{samp}(\mathsf{f}_{\mathsf{src}}(s; \mathscr{F}_T); B), \mathsf{samp}(\mathscr{F}_T - \mathsf{f}_{\mathsf{src}}(s; \mathscr{F}_T); B)\}. \tag{16}$$

Similarly, we construct $\mathscr{D}$ that contains flows received by and not received by same addresses:

$$\begin{aligned} \mathscr{D} &= \bigcup_{\forall d \in \mathcal{D}} \{\mathscr{D}_d\}, \quad \mathscr{D}_d = \{\mathscr{D}^d_{\mathsf{same}}, \mathscr{D}^d_{\mathsf{diff}}\}, \\ \mathscr{D}_d &= \{\mathsf{samp}(\mathsf{f}_{\mathsf{dst}}(d; \mathscr{F}_T); B), \mathsf{samp}(\mathscr{F}_T - \mathsf{f}_{\mathsf{dst}}(d; \mathscr{F}_T); B)\}. \end{aligned} \tag{17}$$

Finally, we denote the training datasets for the contrastive learning as $\mathscr{C} = \mathscr{S} \cup \mathscr{D} = \bigcup_{h \in \mathcal{S} \cup \mathcal{D}} \{\mathscr{C}^h_{\mathsf{same}}, \mathscr{C}^h_{\mathsf{diff}}\}$.

Contrastive Training. Initially, we use tFusion to extract the feature for each flows in a batch $\{\mathscr{C}^h_{\mathsf{same}}, \mathscr{C}^h_{\mathsf{diff}}\}$. That is, we construct the feature matrix ($1 \leq i \leq 2B$):

$$\mathbf{F}^{(h)} = [\vec{F}^1, \ldots, \vec{F}^B, \vec{F}^{B+1}, \ldots, \vec{F}^{2B}], \tag{18}$$

$$\begin{cases} \vec{F}^i = \mathsf{tFusion}(\{\mathscr{C}^h_{\mathsf{same}}\}_i), & \text{if} \quad i \leq B, \\ \vec{F}^i = \mathsf{tFusion}(\{\mathscr{C}^h_{\mathsf{diff}}\}_{i-B}), & \text{else}. \end{cases} \tag{19}$$

Note that tFusion() denotes the overall procedure of our method, as described in Sections 4.1 and 4.2. Afterward, we project the extracted features using two fully connected layers, like the approach by Chen et al. [14]:

$$\mathbf{Q}^{(h)} = \mathsf{proj}(\mathbf{F}^{(h)}) = \mathsf{Linear}\left(\mathsf{ReLU}(\mathsf{Linear}(\mathbf{F}^{(h)}; \mathbf{W}^1, \vec{b}^1)); \mathbf{W}^2, \vec{b}^2\right). \tag{20}$$

Notably, the layers are only used for pre-training, and are not executed during the training phase. We use cosine similarity to measure the distance between two samples:

$$\mathsf{Sim}(\vec{x}, \vec{y}) = \frac{\vec{x}^{\mathsf{T}} \cdot \vec{y}}{\|\vec{x}\| \|\vec{y}\|}. \tag{21}$$

Finally, the loss associated with the batch is:

$$l^{(h)}_{\mathsf{c}} = \frac{1}{B} \sum_{i=1}^{i=B-1} -\ln \left[\frac{e^{\mathsf{Sim}(\vec{q}_i, \vec{q}_{i+1})}}{e^{\mathsf{Sim}(\vec{q}_i, \vec{q}_{i+1})} + e^{\mathsf{Sim}(\vec{q}_i, \vec{q}_{i+B})}} \right]. \tag{22}$$

Once we derive the loss for one batch of pre-training data, we perform back propagation to optimize all the trainable parameters using the Adam optimizer with a small learning rate and weight decay. Unlike existing contrastive learning approaches [14], we do not rely on fixed rules to craft new samples (e.g., image rotation). Instead, we directly utilize flows generated by different addresses with diverse traffic patterns. In addition, our contrastive learning does not rely on labeled datasets, and thus can utilize the rich traffic patterns in the unlabeled large-scale Internet traffic datasets, enabling the attention models to effectively extract features in various network environments.

5 Experiments

We prototype tFusion and compare it with 14 existing detection systems across 11 public datasets, which include 150 different attacks. To complement the existing synthesized datasets, we also deploy tFusion in an institution network. The experimental results demonstrate that:

(1) tFusion outperforms existing methods under various scales of limited datasets (Section 5.2).
(2) Attackers cannot evade tFusion by constructing adversarial examples according to existing strategies (Section 5.3).
(3) tFusion is able to processe high-speed traffic with low latency on a physical testbed (Section 5.4).
(4) tFusion identifies maually constructed attacks, by learning from minimal training samples. (Section 5.5).

5.1 Experiment Setup

Implementation. We prototype tFusion with 2.7K lines of C++14 and Python 3.10 code. We utilize libpcap++ (v22.05) to implement the network component for packet parsing and feature extraction (compiled by GCC v9.4.0, Ninja v1.10.0, and CMake v3.16.3). Meanwhile, we utilize PyTorch (v1.11.0 for CUDA v11.3) to implement the attention models, and use scikit-learn (v1.1.2) to implement the AutoML module.

Testbed. We install the prototype on a DELL server with two Intel Xeon E2699 v4 CPUs, 512GB DDR4 memory, Intel 82599SE NIC (2 × 10Gb/s SFP+ transceivers), and Ubuntu v20.04.2 (Linux v5.15.0). The attention models are trained and executed on a Tesla V100 GPU (32 GB memory, driver v470.103.01). We connect the server to another one with a similar configuration using fiber-optic cables. The other server forwards traffic to this server and coordinates the speeds of network interfaces.

Datasets. We use the following datasets to evaluate tFusion: (i) HyperVision datasets [34], collected from a 10 Gb/s optical link, contain encrypted traffic generated by exploiting real vulnerabilities [47, 62]. (ii) Five datasets collected by CIC that are widely used for evaluating traffic detection systems, including DoH covert channels [30], Android malware [27], stealthy attacks in IoT networks [32], and intrusions in enterprise networks [29, 31]. (iii) Whisper datasets [35] cover reconnaissance steps and advanced attacks, such as link flooding attacks (LFAs) [51] and pulsing attacks [54, 60]. (iv) Kitsune datasets [64] contain attack traffic targeting IoT devices [82]. (v) NetBeacon datasets [107] are collected in a private cloud. (vi) CTU datasets [85] are collected from a campus network. Note that we replay benign traffic collected from the HyperVision datasets when a dataset does not contain benign traffic [29] or contains low-speed simulated traffic [64], to compare the methods in complex network environments. By default, tFusion is pretrained using Internet traffic collected from the backbone network maintained by the MAWI project [94] in Jan. 2023. Specifically, the pretraining dataset contains 1.76 million flows (39 million packets), which are transmitted through fiber-optic cables connecting two ASes in Japan. Detailed network topology and statistics are available on the project website [94].

Metrics. We primarily use AUC (AUROC, the area under the receiver operating characteristic curve) and F1-score (the harmonic mean of precision and recall), because they are widely used in previous work [23, 43, 110]. To prevent biased metric selection [5], we measure other accuracy metrics, including the area under the precision-recall curve (AUPRC), accuracy (Acc.), precision (Pre.), recall (Rec.), true- and false-positive rates (TPR/FPR).

Baselines. We compare tFusion with 14 existing systems, covering both supervised and unsupervised methods. Specifically, we select detection methods which analyze packet features [43, 64, 70], flow features [8, 79, 107], and host features [33, 34]. We deploy open-source methods [33, 34, 64], and prototype closed-source methods [35, 59] and hardware-specific methods [77, 107]. For end-to-end detection, we use Random Forests to learn the CICFlowMeter features [28] and the features extracted by Jaqen [59]. All models are retrained to achieve the highest accuracy. Additionally, we have attempted to adapt benign traffic classification models [58, 104, 105] for detecting malicious traffic. However, none of them achieves acceptable accuracy (above 0.6 AUC), as these methods employ large transformer models which require large training datasets [108].

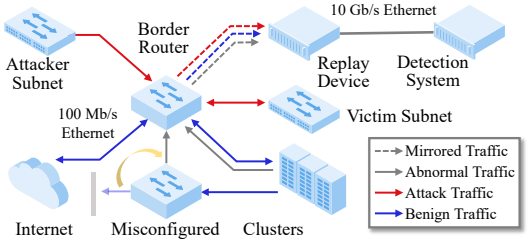


Figure 8: Network topology of deployment.

Deployment. The existing datasets are either synthesized or simulated, which combine traffic from different networks or generated by tools without real users. To mitigate the issues, we present a case study in an institutional network, where we invited human users to investigate the labor consumption required by tFusion.

Figure 8 shows the network topology. We configure a router in the institutional network to mirror the ongoing traffic targeting a subnet to our testbed. The subnet hosts servers are owned by a security research team with around 100 active users. We invite two graduate students (blue team) to train and deploy tFusion. Specifically, they randomly sample 1.0 ‰ flows (91) from the traffic collected in the first 20 minutes (90,319 flows). Afterward, they manually inspect and label the flows using the WireShark GUI. Finally, the team selects 87 benign flows as the training data for the AutoML module to train and select unsupervised models within 10 minutes.

We invite four red teams, each comprising one graduate student and one security expert from the industry. These teams generate attack traffic from a separate subnet, targeting our own victim servers. The conducted attacks include: (i) web attacks: using vulnerability detection tools; (ii) flooding attacks: generating volumetric traffic according to existing studies; (iii) advanced attacks: exploiting protocol vulnerabilities; and (iv) malware behaviors: replaying traffic from sandboxes generated by recently disclosed malware. Note that the router mixes benign and attack traffic and forwards the mixed traffic to the testbed. The ground-truth labels are generated based on whether the traffic originates from the attacker's subnet. Additionally, we observed abnormal traffic generated by real users who were analyzing the prevalence of a recent OpenSSH vulnerability on the Internet.

5.2 Accuracy Evaluation

We compare tFusion with 14 existing methods on 11 datasets. For the fairness of comparison, we restrict the AutoML module to select Random Forest and K-Means models, the widely used supervised and unsupervised models in existing studies [33, 43, 45, 107].

Comparison with Existing Methods. Table 2 compares accuracy when using 10.0% datasets for training. Note that the ratio is significantly lower than using 60% ~ 80% samples for training in previous work [34, 64, 100]. We observe that tFusion achieves stable accuracy across the datasets, i.e., it allows the supervised and unsupervised models to achieve 0.9947 and 0.9783 AUC, thereby outperforming existing methods that achieve the best performances by 9.25% to 12.76%. Moreover, even though existing methods may achieve higher accuracy on some datasets, such as nPrintML [43] and Whisper [33], they cannot achieve high overall accuracy.

Second, when only 1.0 ‰ training samples are available (see Table 3), tFusion can still retain 0.9492 and 0.9864 accuracy, i.e., a decrease of 0.83% and 2.97% in accuracy when comparing with training on 10.0% datasets. In contrast, the accuracy of existing methods decreases by 5.26% ~ 34.97%. Specifically, the accuracy of supervised methods decreases by at least 14.33%. For instance, Taurus using SVM models and achieving superior accuracy, suffers from a 0.129 AUC decrease. Meanwhile, the accuracy of unsupervised methods decreases by at least 5.226%, e.g., a 0.2985 AUC decrease by Hypervision [34], because the limited datasets cannot provide sufficient flow interaction information.

Third, we observe that identifying stealthy attacks with a limited dataset is challenging. tFusion utilizes 21 ~ 194 training samples (10 ~ 176 attack flows), to realize 0.9811 ~ 0.9962 AUC against various

Table 2: Detection accuracy of using 10.0% datasets as training datasets.

Methods	HyperVision Datasets					Datasets Collected by CIC						Datasets in Exisitng Studies				Overall
	Flood	Prob	Web	Malware	Adv.	IDS'17	IDS'18	DoS'19	Android	IoT	DoH	CTU	Whisper	Kitsune	NetB.	
N3IC	0.8572	0.7483	0.8774	0.9492	0.8540	0.9839	0.9884	0.7656	0.7023	0.8506	0.9456	0.9073	0.9211	0.8900	0.8145	0.8766
Taurus	0.9062	0.7435	0.9202	0.9512	0.8605	0.9398	0.9895	0.7648	0.7547	0.8950	0.8907	0.9700	0.9224	0.9841	0.9198	0.9029
FlowMeter-RF	0.9225	0.7541	0.8701	0.9068	0.9471	0.9685	0.9778	0.7849	0.7494	0.8610	0.9261	0.9808	0.9079	0.9747	0.8668	0.8946
Jaqen	0.7155	0.7458	0.9224	0.9448	0.9810	0.9977	0.9993	0.7674	0.8859	0.8825	0.9750	0.7265	0.9557	0.8812	0.6885	0.8778
NetBeacon	0.8588	0.7482	0.8755	0.9497	0.9859	0.9800	0.9905	0.7656	0.7808	0.8959	0.6260	0.6673	0.9254	0.8804	0.8153	0.8587
FlowLens	0.7775	0.7260	0.8707	0.8309	0.8495	0.8965	0.7569	0.5520	0.7381	0.7413	0.8851	-	0.8456	0.7725	0.8217	0.7742
nPrintML	0.9987	0.7937	0.8472	0.9342	0.9144	0.9960	0.8408	0.6497	0.7959	0.9453	0.7892	0.6809	0.9555	0.9973	0.9204	0.8735
RAPIER	0.8604	0.7142	0.8570	0.8875	0.8007	0.9802	0.8173	0.7712	0.6212	0.8895	0.7428	0.7105	0.8079	0.9472	0.7488	0.7972
Kitsune	0.6069	0.6999	0.6823	0.5677	0.7790	0.6641	0.6009	0.6478	0.8472	0.6966	0.9924	0.9970	0.6810	0.6645	0.6078	0.7025
FSC	0.8791	0.6736	0.6545	0.7484	0.8081	0.5690	0.7007	0.7754	0.9075	0.6933	0.9451	0.6949	0.8525	0.7906	0.8653	0.7690
Whisper	0.7842	0.9248	0.9617	0.5375	0.7451	0.5952	0.6079	0.7902	0.5246	0.8418	0.9980	0.5147	0.9186	0.8896	0.7753	0.7523
FlowMeter-KM	0.8440	0.6668	0.7495	0.7255	0.8694	0.8049	0.7588	0.6619	0.8605	0.8171	0.9644	0.5726	0.7746	0.7482	0.8024	0.7727
FAE	0.7812	0.9245	0.9478	0.5377	0.7396	0.5924	0.6070	0.7897	0.5247	0.8310	0.9470	0.5147	0.9134	0.8826	0.7712	0.7459
Hypervision	0.8112	0.9832	0.9664	0.9085	0.8440	0.9903	0.9990	0.9993	0.6197	0.8360	0.5287	0.6205	0.8824	0.6373	0.8850	0.8534
tFusion-RF	0.9999	1.0000	1.0000	1.0000	0.9961	1.0000	1.0000	1.0000	1.0000	0.9344	0.9857	0.9976	1.0000	1.0000	0.9999	0.9947
tFusion-KM	0.9851	0.9763	0.9775	0.9684	0.9760	0.9858	0.9900	0.9875	0.9622	0.9063	0.9952	0.9950	0.9839	0.9843	0.9893	0.9783

Table 3: Detection accuracy of using 1.0 ‱ datasets as training datasets.

Methods	HyperVision Datasets					Datasets Collected by CIC						Datasets in Exisitng Studies				Overall
	Flood	Prob	Web	Malware	Adv.	IDS'17	IDS'18	DoS'19	Android	IoT	DoH	CTU	Whisper	Kitsune	NetB.	
N3IC	0.6860	0.7487	0.7483	0.8044	0.6995	0.6961	0.7651	0.8955	0.6592	0.6960	0.9275	-	0.7538	0.9198	0.6527	0.7333
Taurus	0.7703	0.7273	0.7585	0.8610	0.7709	0.7177	0.6031	0.5385	0.8190	0.8029	0.9247	0.7377	0.8902	0.9319	0.7918	0.7735
FlowMeter-RF	0.7676	0.7610	0.8081	0.7255	0.7152	0.7383	0.6715	0.7586	0.6835	0.7769	0.9446	-	0.7494	0.8784	0.7509	0.7418
Jaqen	0.6925	0.7284	0.8640	0.7682	0.7120	0.6802	0.8090	0.9092	0.5815	0.8615	0.7277	-	0.8052	0.8371	0.6978	0.7382
NetBeacon	0.7343	0.6938	0.8451	0.8046	0.7215	0.6583	0.6023	0.7691	0.6598	0.8755	0.7203	-	0.7766	0.7176	0.7160	0.7121
FlowLens	0.5012	0.5008	0.5428	0.5294	-	-	-	-	-	-	0.5832	-	0.5004	-	0.5011	0.5093
nPrintML	0.9217	0.7196	0.8145	0.8413	0.7535	0.5555	0.6418	0.6461	0.6083	0.7945	0.7221	-	0.7499	0.8789	0.8776	0.7385
RAPIER	0.5119	0.6709	0.5923	0.6101	0.6291	-	-	0.5268	-	0.5765	0.7324	-	0.6482	0.8418	0.5072	0.5792
Kitsune	0.7703	-	-	-	0.6441	-	-	-	-	0.5679	-	-	0.6302	0.5525	0.6122	0.5457
FSC	0.7857	-	0.6161	0.5001	0.7732	-	0.6538	0.6414	0.8218	0.7038	0.8724	0.7594	0.7633	0.6504	0.7475	0.6783
Whisper	0.7809	0.8923	0.8857	0.5375	0.7499	0.5929	0.5864	0.7418	0.5244	0.7904	0.9655	0.5143	0.8809	0.5559	0.7724	0.7127
FlowMeter-KM	0.7841	-	0.5660	0.5171	0.7605	-	0.5971	0.5971	0.7175	0.6411	0.8577	-	0.6580	0.6797	0.7028	0.6299
FAE	0.7779	0.5236	0.8839	0.5308	0.6778	0.5947	0.5897	0.5490	0.5244	0.7838	0.7329	0.5071	0.7340	-	0.7684	0.6484
Hypervision	0.5397	0.5594	0.5872	0.5669	0.6245	0.5702	0.5351	0.5328	0.5448	0.5722	-	-	0.5775	0.5791	0.5409	0.5549
tFusion-RF	0.9963	0.9787	0.9877	0.9506	0.9888	0.9966	0.9960	0.9792	0.9565	0.9809	0.9922	0.9726	0.9975	0.9997	0.9973	0.9864
tFusion-KM	0.9445	0.8279	0.9847	0.9262	0.8829	0.9912	0.9937	0.9890	0.9855	0.9183	0.9713	0.9972	0.9316	0.9665	0.9572	0.9492

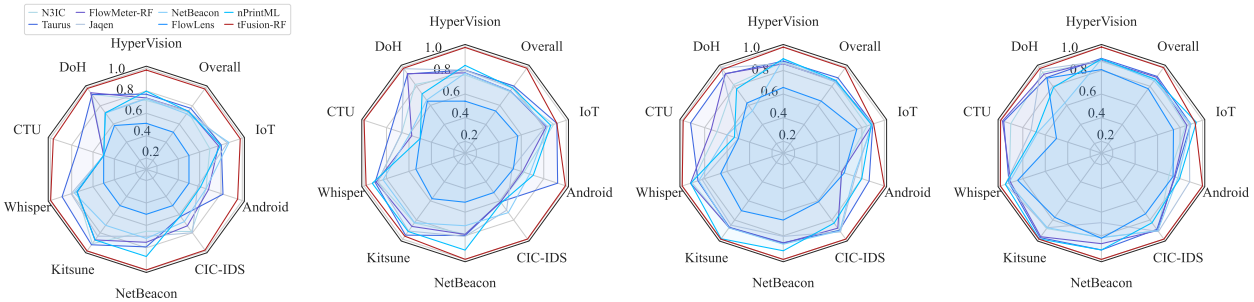
[1] NetB. stands for NetBeacon datasets [107]; Adv. is short for stealthy attacks in HyperVision datasets [34]; RF and KM denote Random Forest and K-Means.
[2] The - indicates that the performance is not better than random guessing.

stealthy network probings, web vulnerabilities, and malware infections. Particularly, tFusion captures advanced attacks with 0.9760 average AUC, including the Crossfire attack [51], side-channel attack (CVE-2020-36516 [26]), TCP hijacking attacks (CVE-2016-5696 [13]), and pulsing TCP DoS attacks [60].

Comparing Different Ratios of Samples. Figure 9 reports the accuracy for different ratios of samples for training. Specifically, tFusion enables supervised models to achieve 0.9947 ∼ 0.9864 AUC when using 1.0 ‱, 1.0 ‰, 1.0%, and 10% training datasets. Compared to the supervised baselines, tFusion improves their accuracy by over 17.79% ∼ 8.20%. Similarly, we compare the accuracy for unsupervised models in Figure 10. We find that the accuracy of tFusion ranges between 0.9675 and 0.9784, thereby significantly outperforming 0.8093 ∼ 0.8489 accuracy achieved by the baselines.

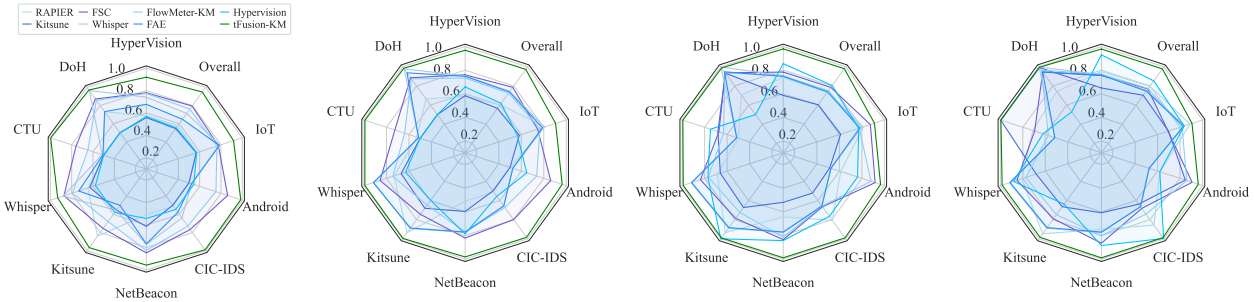
We observed that training with more samples incurs the overfitting issue [65], i.e., the model memorizes redundant training samples, reducing its accuracy in classifying testing samples [19]. Our experiments confirmed the overfitting issue: the accuracy on training samples was 8.16% higher than on test samples. In addition, we compare the accuracy using different numbers of samples. tFusion achieves 0.9863 ∼ 0.9946 accuracy with 50 ∼ 200 training samples. These details can be found in Appendix C.1.

Comparing Data Augmentation Methods. Existing approaches generate more data using fixed-rules or Generative Adversarial Network (GAN), which are applicable to security tasks other than traffic detection, such as website fingerprinting (WF) attacks [6, 48] and fake user detection [46]. We adapt five data augmentation methods for comparison (see Appendix B.2 for configurations). Figure 11 shows the results of the comparison. First, we observe that the rule based augmentation strategies (NetAugment and Rosetta) achieve at most 0.8895 AUC, which is significantly lower than tFusion. Since their rules are designed for Tor and TLS traffic based



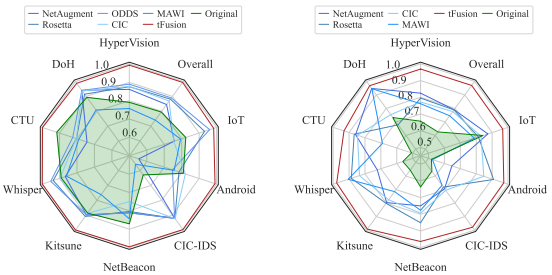
(a) 1.0 ‱ samples for training. (b) 1.0 ‰ samples for training. (c) 1.0% samples for training. (d) 10.0% samples for training.

Figure 9: tFusion enables supervised models to detect various attacks when using different ratios of samples for training.



(a) 1.0 ‱ samples for training. (b) 1.0 ‰ samples for training. (c) 1.0% samples for training. (d) 10.0% samples for training.

Figure 10: tFusion enables unsupervised models to detect various attacks when using different ratios of samples for training.



(a) Supervised models (50 samples). (b) Unsupervised models.

Figure 11: Accuracy of existing methods with augmentations.

on prior knowledge, they cannot effectively augment other types of traffic. Moreover, GAN based ODDS achieves 0.7404 AUC, because it can only generate malicious features. Furthermore, by adding the datasets, the supervised model achieves at most 0.8813 AUC, as the feature distributions of testing datasets are different from the complemented ones. Overall, when applying these methods to supervised detection, their accuracy is 10.52% ∼ 25.52% lower than tFusion. In addition, the methods achieve 0.7460 ∼ 0.7911 AUC based on unsupervised methods, which is 18.45% ∼ 23.10% lower.

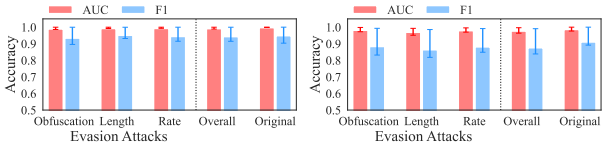
Comparison for Ablation Study. We replace our cross-attention traffic feature fusion model with simple fusion methods. Particularly, we use addition and concatenation instead of using our cross-modal attention to fuse the multimodal features. tFusion outperforms the concatenation fusion by over 8.06% accuracy. Meanwhile, the addition method can only achieve 87.65% of the AUC compared to tFusion. In addition, we observe that using solely packet, flow, and host features can only achieve an AUC ranging between 0.8508 and 0.5409, which is at most 44.09% lower than using the fused traffic features. Moreover, we individually disable the three kinds of features, i.e., only two modalities are involved in the multiplication. We observe that the absence of the features leads to an 8.50% to 14.70% AUC decrease for supervised models and a 3.79% to 47.48% AUC decrease for unsupervised models. Thus, we conclude that effective detection with minimal datasets can only be achieved by simultaneously using all the three modalities.

Furthermore, from Figure 25, we observe that the AutoML module can improve 3.53% accuracy by accurately selecting ML models with best performances. Meanwhile, it outperforms random model selection by 3.24% accuracy. In Appendix C.2, we validate other design choices, e.g., packet embedding and contrastive learning.

5.3 Robustness Evaluation

Robustness Against Evasion Attacks. Unlike traditional methods [41–43], tFusion does not directly use raw bytes as input features. Thus, it cannot be easily evaded by tampering payloads [44]. Therefore, we construct adversarial examples according to recent studies on feature manipulation [33, 35, 70]: (i) Traffic obfuscation: attackers inject benign TCP/UDP encrypted traffic to obfuscate attack traffic at a ratio of 1:4; (ii) Sending rate reduction: attackers decrease their sending rates by 50%; (iii) Length manipulation: attackers mimic benign encrypted flows by manipulating packet lengths according to 5.0% randomly selected benign flows. According to the strategies, we generate 48 evasion attack based on 16 public traffic datasets [34].

From the results in Figure 12(a), we can see that the accuracy decrease incurred by the evasion attacks can be bounded by 0.80% AUC and 1.54% F1. Overall, tFusion allows random forest models to achieve 0.9927 AUC and 0.9445 F1, which is 0.59% and 0.58% lower than the accuracy without the interference of evasion attacks. Moreover, for unsupervised models, the evasion attacks can



(a) Supervised models. (b) Unsupervised models.

Figure 12: Detection accuracy under evasion attacks.

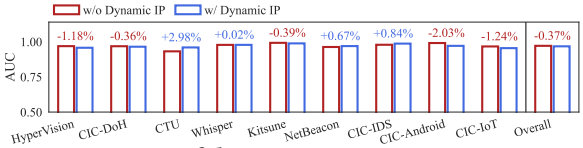


Figure 13: Impact of dynamic IP assignment on accuracy.

decrease AUC by at most 0.46% ∼ 0.94%. Note that the accuracy drop is lower than of existing non-robust methods, such as 35.40% decrease [64], and is similar to existing robust detection, e.g., 3.67% drop by Whisper [35] and 4.49% drop by HyperVision [34]. The reason for such robust detection is that existing evasion attacks exclusively manipulate sequential or non-sequential features, which deviate from the normal relationship among the features. Thus, tFusion can captures the evasion attack by correlating the different granulates of feature using the crossmodal attention model.

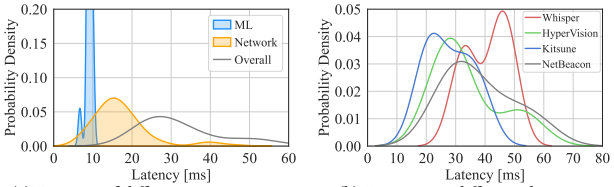
Robustness Against Dynamic IP. In addition, we analyze the impact of dynamic IP assignment on our topology-driven contrastive learning method. Specifically, we utilize the probabilistic model for IP assignment [12] to simulate dynamic environments. That is, when replaying the traffic datasets, we randomly change the IP addresses of packets according to the probability of IP reassignment. Figure 13 shows that tFusion retains over 97.9% accuracy, when unsupervised ML models are trained with 50 samples from different datasets. Therefore, the impact of dynamic IP assignment is limited, since existing works [12, 68] show that Internet IPs are rarely reassigned during short time windows, e.g., a few minutes of data collection for pretraining traffic datasets [94].

Other Robustness Analysis. In Appendix D, we consider other adversarial techniques, such as topology manipulation. Moreover, we evaluate the hyperparameter sensitivity, validate the stability of pre-training using various traffic datasets, and measure the robustness with respect to different models.

5.4 Performance Evaluation

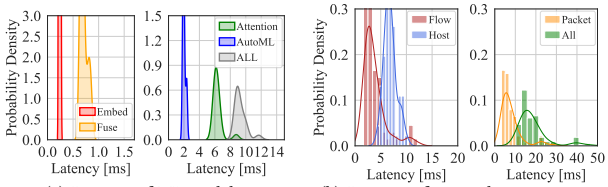
We measure efficiency of the ML models and network component of tFusion. In general, it achieves efficient detection with 32.23ms latency and 0.7042 million packet per second (MPPS) throughput.

Detection Latency. Figure 14(a) plots the probability distribution function (PDF) of detection latency on the HyperVision dataset. The average latency of the ML models and network components is 9.02ms and 17.22ms, respectively. The latency of the network component is higher due to the scale of Internet packets. Figure 14(b) shows the data about the latency for different datasets. The average latency on Whisper, HyperVision, Kitsune, and NetBeacon datasets is 40.70ms, 34.21ms, 27.89ms, and 38.17ms, respectively. Moreover, we analyze the composition of the latency in Figure 15. The embedding module, sequence model, feature fusion, and AutoML module exhibit 0.23ms, 6.34ms, 0.79ms, and 2.02ms latency, respectively. In addition, packet, flow, and host feature extractions require 7.75ms, 3.76ms, and 6.81ms on average.



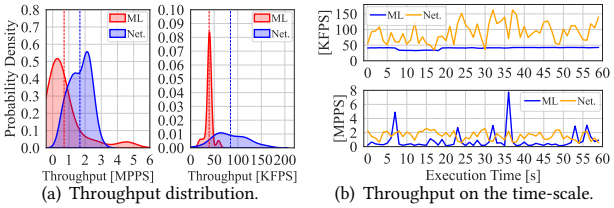
(a) Latency of different components. (b) Latency on different datasets.

Figure 14: Latency of the components on different datasets.



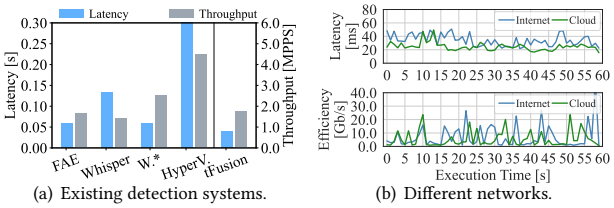
(a) Latency of ML models. (b) Latency of network components.

Figure 15: Composition of tFusion detection latency.



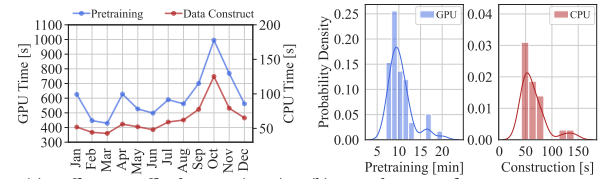
(a) Throughput distribution. (b) Throughput on the time-scale.

Figure 16: Detection throughput of different components.



(a) Existing detection systems. (b) Different networks.

Figure 17: Comparing efficiency with existing methods.



(a) Different traffic datasets (2023). (b) Distributions of time consumption.

Figure 18: Overheads of pretraining with Internet traffic.

Detection Throughput. Figure 16(a) shows the data on the throughput of tFusion. We can see that the ML models and network component can process 0.70 and 1.68 million packets per second (MPPS), respectively. Thus, the capacity of tFusion is higher than the throughput of high-speed ISP networks, e.g., 0.28 MPPS [94], thereby achieving low-latency detection in high-speed networks. In addition, the network component can process 84.50 thousand flows per second (KFPS) and the ML models achieve 39.23 KFPS throughput. Note that such performance can be scaled up by running multiple instances of tFusion. Moreover, the data in Figure 16(b) show that tFusion achieves stable performance and the ML models can achieve 7.75 MPPS maximum efficiency.

Table 4: Detection accuracy in the network environment using 1.0 ‰ of the captured flows as training dataset.

Groups	Abnormal Behaviors	Times			Statistics		Speeds		Accuracy Metrics					
		Start	End	Span	Packets	Flows	KPPS	Mb/s	AUROC	AUPRC	F1	Acc.	Pre.	Rec.
Web Attack	Scrapy Clawer	17:33	17:40	455.77	116.31K	170	0.2552	1.9892	0.9976	0.9602	0.9630	0.9966	0.9278	0.9982
	CSRF Detection	21:10	21:15	329.77	50.43K	887	0.1529	0.8378	0.9975	0.9907	0.9908	0.9966	0.9836	0.9981
	SSL Detection	08:14	08:15	88.58	11.98K	1466	0.1352	0.3266	0.9975	0.9956	0.9956	0.9977	0.9926	0.9986
	XSS Detection	10:12	10:13	91.43	14.66K	174	0.1603	1.0998	0.9975	0.9637	0.9660	0.9968	0.9337	0.9984
	SQL Injection	11:34	11:36	128.79	21.91K	1338	0.1701	0.4644	0.9979	0.9920	0.9921	0.9960	0.9865	0.9977
	Nginx Injection	15:57	16:01	284.78	68.94K	1370	0.2421	1.7089	0.9977	0.9960	0.9960	0.9980	0.9931	0.9988
	Spam Flooding	17:39	17:50	660.31	110.61K	282	0.1675	1.6358	0.9975	0.9718	0.9732	0.9961	0.9484	0.9980
Flooding Attack	Crossfire LFA	20:29	20:51	1350.77	395.07K	2654	0.2925	2.0957	0.9974	0.9964	0.9964	0.9973	0.9946	0.9982
	Pulsing TCP DoS	22:12	22:27	952.47	517.74K	54	0.5436	6.2035	0.9972	0.9990	0.9990	0.9999	0.9999	0.9981
	Coremelt LFA	8:03	08:15	764.78	225.66K	28	0.2951	2.5186	0.9976	0.9990	0.9990	0.9998	0.9999	0.9980
	SYN Flooding	10:00	10:00	14.80	5.88K	1726	0.3972	0.1398	0.9974	0.9953	0.9953	0.9972	0.9923	0.9983
	Flash Crowd	11:04	11:06	122.01	53.07K	107	0.4349	4.9688	0.9975	0.9245	0.9341	0.9956	0.8705	0.9977
	NTP Amplification	16:21	16:21	18.48	160.97K	5630	0.3045	0.1851	0.9974	0.9978	0.9978	0.9979	0.9974	0.9982
	SSH Bug Probing (1)	17:26	17:38	749.79	107.52K	17897	0.1434	0.1094	0.9975	0.9988	0.9988	0.9990	0.9992	0.9983
	Active Host	19:59	19:59	44.70	21.63K	1690	0.4838	0.1554	0.9976	0.9952	0.9952	0.9972	0.9921	0.9983
	CLDAP Amplification	20:37	20:37	6.68	3.38K	845	0.5056	0.3236	0.9976	0.9983	0.9983	0.9987	0.9992	0.9975
	DNS Probing	22:41	22:42	63.79	24.00K	29	0.3762	0.4536	0.9976	0.9716	0.9731	0.9996	0.9464	0.9998
Advanced Attack	Side-Channel DNS	8:57	08:59	127.75	35.80K	1645	0.2802	0.0897	0.9976	0.9927	0.9928	0.9959	0.9881	0.9975
	SSH Bug Probing (2)	9:30	10:09	2349.80	240.47K	130	0.1023	0.1086	0.9975	0.9524	0.9564	0.9968	0.9145	0.9984
	Side-Channel IPID	10:21	10:25	253.72	57.31K	31	0.2259	0.0653	0.9975	0.9821	0.9827	0.9998	0.9655	0.9999
	Side-Channel ACK	10:45	10:46	95.51	19.35K	4255	0.2026	0.0722	0.9979	0.9969	0.9970	0.9972	0.9960	0.9979
	Password Cracking	11:25	11:34	575.80	63.09K	117	0.1096	0.1515	0.9972	0.9986	0.9986	0.9996	0.9998	0.9975
	Padding Oracle	15:53	15:55	134.77	45.56K	30	0.3380	3.8943	0.9967	0.9992	0.9992	0.9998	0.9999	0.9985
	Telnet Injection	16:53	17:07	873.79	99.00K	10	0.1133	0.0363	0.9975	0.9993	0.9993	0.9999	0.9999	0.9987
	Lateral Movement	17:42	18:06	1471.80	156.11K	5708	0.1061	0.6400	0.9977	0.9974	0.9974	0.9974	0.9970	0.9978
	SSH Bug Probing (3)	22:07	22:16	579.82	57.58K	646	0.0993	0.0488	0.9974	0.9987	0.9987	0.9995	0.9997	0.9977
Malware	Mirai Malware	8:30	08:50	1209.80	200.93K	28672	0.1661	0.0531	0.9977	0.9988	0.9988	0.9992	0.9995	0.9982
	Sality Malware	10:17	10:36	1169.45	155.16K	64128	0.1327	0.0605	0.9974	0.9988	0.9988	0.9995	0.9997	0.9978
	Topology Change	11:19	11:27	530.77	378.42K	16452	0.7130	4.7164	0.9977	0.9990	0.9990	0.9991	0.9993	0.9986
All	Overall	-	-	534.49	117.88K	5454.17	0.2637	1.2121	0.9975	0.9882	0.9890	0.9980	0.9798	0.9982

[1] The shade ■ indicates unseen abnormal traffic generated by real users, and ■ indicates the network reconfiguration event.
[2] We highlight the best accuracy in ● and the worst accuracy in ●.

Efficiency Comparison. In Figure 17(a), we compare the efficiency with existing realtime methods on the high-speed network testbed. We find that the overall latency of tFusion is lower than existing methods. In specific, the latency of FAE [35], HyperVision [34], Whisper [33] w/o and w/ sampling is 1.41, 26.77, 3.28, and 1.42 times higher, because tFusion can utilize GPUs for efficient feature extraction. Moreover, the throughput of tFusion is comparable to that of existing methods. In particular, its throughput is 1.05 and 1.23 times higher than FAE and Whisper. HyperVision has higher efficiency yet incurs 0.91s high latency. Furthermore, from Figure 17(b), we observe that the performances in different networks are similar. That is, tFusion achieves 34.89ms latency and 5.17 Gb/s throughput in the Internet environment [94], which is similar to 26.45ms latency and 6.52 Gb/s throughput in the cloud [107].

Pretraining Overheads. Our pretraining takes 10.43 minutes using a single NVIDIA Tesla V100 GPU, where the average GPU utilization is 13.5%. Such overhead is comparable to those reported in related studies [17, 37, 70], e.g., training Transformers for traffic analysis requires 30 minutes [17]. Moreover, Figure 18 compares the overheads using different datasets collected in different months [94]. The time consumption ranges between 7.14 ~ 16.58 minutes with an average of 10.18 minutes. Furthermore, it takes 63.67s to pre-process datasets, i.e., to construct contrastive data pairs for pretraining. Note that pretraining overheads on some datasets are relatively higher (e.g., Aug. 2023), as the datasets contain more flows.

5.5 Deployment

Analyzing the Statistics of Traffic. During the deployment, tFusion processes 25.43M flows (884.25M packets), where 4.823% flows (3.263M packets) are associated with abnormal behaviors. We plot the speed of traffic in Figure 19, which ranges between 0.279K and 4.588K packets per second (PPS). Note that the average speed of attack traffic is only 0.008K PPS, which is significantly lower, due to the stealthy attacks. Moreover, we observe that the subnet is accessed by 62 ~ 140 active users (i.e., the users send over 20 flows in an hour). In addition, we analyze the flow and packet distributions. The associated observations are similar to the ones reported in previous work [34] (see Figure 21 in Appendix B.3).

Accuracy During the Deployment. From Table 4, we observe that tFusion achieves 0.9975 AUROC, 0.9882 AUPRC, 0.9798 Precision, and 0.9982 Recall on average. Specifically, tFusion can detect web attacks, flooding attacks, advanced vulnerability exploits, and malware traffic by achieving 0.9830 ~ 0.9988 AUPRC. In particular, tFusion captures the attacks that generate few flows with 0.9821 ~

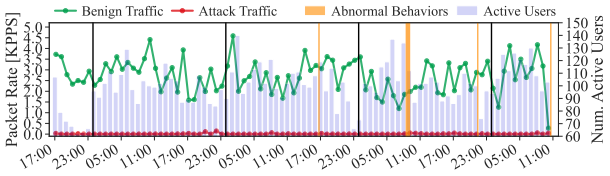


Figure 19: Traffic statistics of the real traffic dataset.

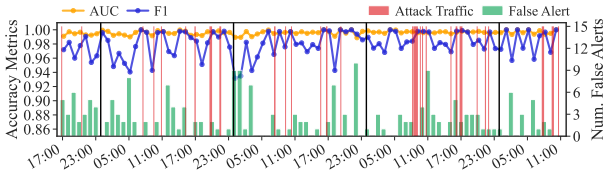


Figure 20: Detection accuracy in the network environment.

0.9990 AUPRC; for example, 10 ~ 54 flows generated Pulsing TCP DoS attacks [60], TCP hijacking attacks [26], and Telnet command injections. In addition, tFusion achieves 0.9718 ~ 0.9986 AUPRC, when detecting low-speed stealthy attack traffic, such as password cracking (109 PPS), SMTP-over-SSH spam (167 PPS), and SQL injections (170 PPS). Furthermore, it identifies attacks with both short and long durations, e.g., detecting short-term flooding attacks with 0.9953 ~ 0.9983 AUROC, and long-term malware activities lasting 20.16 ~ 1471.80 minutes. Note that all the attacks are treated as unseen attacks during the deployment, as our training datasets only contain benign samples.

Identifying Abnormal Behaviors from Real Users. Surprisingly, we discovered that tFusion can identify abnormal traffic generated by real users and administrators. Specifically, it detected traffic generated by a script that probed for a recently disclosed OpenSSH vulnerability at 17:27 on the third day and at 9:30 and 22:07 on the fourth day. tFusion detected the three abnormal behaviors with 0.9524 ~ 0.9988 AUPRC. We contacted the research team to confirm the activities. On the last day, our device identified unusual traffic routed from unseen subnets, after the administrators changed the routing policy, as illustrated in Figure 8. To prevent the unseen traffic from congesting links, we terminated the experiment to preserve link capacity.

False Alerts Issues. We observe that human users can effectively respond to raised alarms by filtering false alarms. Specifically, we plot the number of false alerts in Figure 19. Like previous work [36], tFusion aggregates the flows associated with raised alerts according to the distances in the traffic feature space. Overall, tFusion raises 2.322 false alerts per hour, which falls within the manual processing capability indicated by previous work [24, 36, 40]. In addition, the blue team can easily handle these false alerts since most of these alerts are triggered by repetitive probing traffic and plain-text DNS queries. Furthermore, we validate that tFusion is not significantly affected by the concept drift problem, as the AUROC and F1 score remain over 0.9803 and 0.9317 over five days without a surge in FP numbers.

6 Limitations and Future Works

First, we have validated the robustness of tFusion only against known evasion attacks from previous works [33, 70]. However, as it is challenging to manipulate statistical traffic characteristics while retaining the effectiveness of attacks, we expect novel successful evasion strategies against tFusion to be developed in the future. This will open new research directions in novel attacks and related defenses.

Second, we have carried out some initial experiments to apply tFusion multimodal strategy to another traffic-related task, that is, classifying whether encrypted traffic is generated by particular benign applications. For this application, we have modified the supervised models of AutoML for multiple classes without modifying other modules. Our experiments show that by using just 100 samples to classify VPN and Tor flows [104], tFusion outperforms state-of-the-art methods YaTC [105] and ET-BERT [58] by 2.75 ~ 5.75% F1 and 3.67 ~ 6.62% F1, respectively. Based on these preliminary results, an interesting research direction for future work is to explore multimodal ML for security tasks other than malicious traffic detection.

7 Related Work

ML Based Malicious Traffic Detection. Existing generic detection approaches use a variety of features to simultaneously capture various malicious traffic: (i) Flow features, such as discrete distribution features [8], statistical features [28], sequential features [70], and frequency features; (ii) Packet features, such as Kitsune [64] and nPrintML [43]. Moreover, task-specific detection aims to capture the behaviors of malware [11, 21, 81], web attacks [9, 55, 80], threats in IoT networks [74, 82], and flooding campaigns [73, 99].

In particular, high-speed network devices are leveraged for efficient detection. For example, programmable switch-based systems support high-speed ML inference, e.g., decision trees [45, 107], forests [22], and RNNs [100]. Similarly, SmartNICs with many embedded cores enable flexible deployment [67, 77]. In addition, there are approaches for implementing detection models on FPGAs [37, 79].

Traffic Based Defense Systems. Once abnormal traffic is detected, these systems throttle the abnormal traffic by using fixed rules. SDN based methods support automatic rule deployment [25, 52, 106]. Recent approaches implement complex defense behaviors on programmable switches, e.g., register based Poseidon [103], Sketch based Jaqen [59], Mew [109] using multiple devices, Ripple distributed defense [97], ACC-Turbo for pulsing attacks [4], and NetWarden for covert channels [96]. Many practical defense strategies are designed for traditional forwarding devices, e.g., AS-level defense for ISPs [84] and IXP-level defenses [89, 93]. Additionally, Li *et al.* [57] analyzed the benefits of the defense using game theory.

Attention Models for Security and Privacy Tasks. Attention models are promising for analyzing sequential data. For security-related tasks, these models were initially used to learn instructions for binary analysis [56, 61]. More recently, by viewing logs as sequences and using Transformers, these models have been applied to attack detection [69], forensic [20], and investigation of security incidents [86]. Furthermore, attention models have been designed to discover and mitigate privacy issues, such as web privacy leakage [17, 48, 71, 76].

Approaches to Address Training Data Scarcity. To address the disparity between simulated and real-world datasets, approaches have been proposed by which rules are manually designed to augment Tor traffic for WF attacks [6, 48] and TLS traffic for application

identification [95]. Similarly, Jan *et al.* [46] generated the attack samples for fake user detection using GANs [102]. Moreover, netUnicorn [10] and ISDC [63] are tools for collecting flow datasets. The approach by Qing *et al.* [70] addresses incorrect label issues, and the one by Du *et al.* [24] enables efficient retraining. In addition, the few-, single-, and zero-shot learning paradigms enable supervised ML to classify one newly added class, e.g., meta-learning approaches [55], which are not our potential solutions, as training in new networks encounters numerous classes of new applications and unknown attacks.

Issues in ML-Based Security Applications. Sommer *et al.* [78] analyzed the low usability issues of ML-based intrusion detection systems, and emphasized the prohibitive cost of building new datasets. Arp *et al.* [5] investigated practical issues in applying ML to security [64]. Moreover, Alahmadi *et al.* [3], Vermeer *et al.* [88], and Fu *et al.* [36] found that ML-based security applications raised numerous false alerts. Furthermore, past work analyzed concept drifting issues [7, 50, 101], explainability issues Han *et al.* [40], Jacobs *et al.* [44], and Wei *et al.* [91], sampling bias issues [83] and theoretical stability of intrusion detection systems Wang *et al.* [90].

8 Conclusion

This paper presents tFusion, a malicious traffic detection system, able to effectively extract features from small training datasets using the relationships among various modalities of traffic features. First, tFusion extracts fine-grained packet-level spatial and temporal features. Second, tFusion fuses these fine-grained features with coarse-grained flow and host features through a crossmodal attention mechanism. In addition, the paper introduces a novel topology-driven contrastive learning method for pre-training. Our experiments demonstrate that tFusion enables experts to have to label only 1.0 ‰ of traffic, yet it achieves an accuracy of 99.82% against various real-world attacks. Furthermore, it outperforms 14 existing methods by improving accuracy by more than 12. 76% in 11 datasets.

Acknowledgments

This work was supported in part by the National Science Foundation for Distinguished Young Scholars of China under No. 62425201, the Science Fund for Creative Research Groups of the National Natural Science Foundation of China under No. 62221003, the Key Program of the National Natural Science Foundation of China under No. 61932016 and No. 62132011, the Tsinghua Scholarship for Overseas Graduate Studies under No. 2024072, and the State Key Laboratory of Internet Architecture. (Corresponding author: Ke Xu.)

References

[1] AKamai. Accessed Apr. 2025. Akamai Unveils Machine Learning That Intelligently Automates Application And API Protections And Reduces Burden On Security Professionals. https://www.akamai.com/newsroom/press-release/-akamai-unveils-machine-learning-that-intelligently-automates-ap.
[2] AKamai. Accessed Apr. 2025. Prolexic. https://www.akamai.com/products/prolexic-solutions.
[3] Bushra A. Alahmadi et al. 2022. 99% False Positives: A Qualitative Study of SOC Analysts' Perspectives on Security Alarms. In *Security*. USENIX, 2783–2800.
[4] Albert Gran Alcoz et al. 2022. Aggregate-based congestion control for pulse-wave DDoS defense. In *SIGCOMM*. ACM, 693–706.
[5] Daniel Arp et al. 2022. Dos and Don'ts of Machine Learning in Computer Security. In *Security*. USENIX, 3971–3988.
[6] Alireza Bahramali et al. 2023. Realistic Website Fingerprinting By Augmenting Network Traces. In *CCS*. ACM, 1035–1049.
[7] Federico Barbero et al. 2022. Transcending TRANSCEND: Revisiting Malware Classification in the Presence of Concept Drift. In *SP*. IEEE, 805–823.
[8] Diogo Barradas et al. 2021. FlowLens: Enabling Efficient Flow Classification for ML-based Network Security Applications. In *NDSS*. ISOC.
[9] Karel Bartos et al. 2016. Optimized Invariant Representation of Network Traffic for Detecting Unseen Malware Variants. In *Security*. USENIX, 807–822.
[10] Roman Beltiukov et al. 2023. In Search of netUnicorn: A Data-Collection Platform to Develop Generalizable ML Models for Network Security Problems. In *CCS*. ACM, 2217–2231.
[11] Leyla Bilge et al. 2012. Disclosure: detecting botnet command and control servers through large-scale NetFlow analysis. In *ACSAC*. ACM, 129–138.
[12] Leon Böck et al. 2023. How to Count Bots in Longitudinal Datasets of IP Addresses. In *NDSS*. ISOC.
[13] Yue Cao et al. 2016. Off-Path TCP Exploits: Global Rate Limit Considered Dangerous. In *Security*. USENIX, 209–225.
[14] Ting Chen et al. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML (Proceedings of ML Research, Vol. 119)*. PMLR, 1597–1607.
[15] Cisco. Accessed Apr. 2025. Cisco SPAN. https://www.cisco.com/c/en/us/support/docs/swit-ches/catalyst-6500-series-switches/10570-41.html.
[16] Cloudflare. Accessed Apr. 2025. Cloudflare DDoS Protection Products. https://developers.cloudflare.com/ddos-protection/managed-rulesets/adaptive-protection/.
[17] Xinhao Deng et al. 2023. Robust Multi-tab Website Fingerprinting Attacks in the Wild. In *SP*. IEEE, 1005–1022.
[18] Jacob Devlin et al. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*. ACL, 4171–4186.
[19] Thomas G. Dietterich. 1995. Overfitting and Undercomputing in Machine Learning. *ACM Comput. Surv.* 27, 3 (1995), 326–327.
[20] Hailun Ding et al. 2023. AIRTAG: Towards Automated Attack Investigation by Unsupervised Learning with Log Texts. In *Security*. USENIX, 373–390.
[21] Priyanka Dodia et al. 2022. Exposing the Rat in the Tunnel: Using Traffic Analysis for Tor-based Malware Detection. In *CCS*. ACM, 875–889.
[22] Yutao Dong et al. 2023. HorusEye: A Realtime IoT Malicious Traffic Detection Framework using Programmable Switches. In *Security*. USENIX, 571–588.
[23] Min Du et al. 2017. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. In *CCS*. ACM, 1285–1298.
[24] Min Du et al. 2019. Lifelong Anomaly Detection Through Unlearning. In *CCS*. ACM, 1283–1297.
[25] Seyed Kaveh Fayaz et al. 2015. Bohatei: Flexible and Elastic DDoS Defense. In *Security*. USENIX, 817–832.
[26] Xuewei Feng et al. 2020. Off-Path TCP Exploits of the Mixed IPID Assignment. In *CCS*. ACM, 1323–1335.
[27] Canadian Institute for Cybersecurity. Accessed Apr. 2025. Android malware dataset. https://www.unb.ca/cic/datasets/andmal2017.html.
[28] Canadian Institute for Cybersecurity. Accessed Apr. 2025. CICFlowMeter: a network traffic flow generator and analyser. https://www.unb.ca/cic/research/applications.html.
[29] Canadian Institute for Cybersecurity. Accessed Apr. 2025. DDoS Evaluation Datasets (CIC-DDoS2019). https://www.unb.ca/cic/datasets/ddos-2019.html.
[30] Canadian Institute for Cybersecurity. Accessed Apr. 2025. DNS-over-HTTP datasets. https://www.unb.ca/cic/datasets/dohbrw-2020.html.
[31] Canadian Institute for Cybersecurity. Accessed Apr. 2025. Intrusion Detection Evaluation Datasets (CIC-IDS2017). https://www.unb.ca/cic/datasets/ids-2017.html.
[32] Canadian Institute for Cybersecurity. Accessed Apr. 2025. A real-time dataset and benchmark for large-scale attacks in IoT environment. https://www.unb.ca/cic/datasets/iotdataset-2023.html.
[33] Chuanpu Fu et al. 2021. Realtime Robust Malicious Traffic Detection via Frequency Domain Analysis. In *CCS*. ACM, 3431–3446.
[34] Chuanpu Fu et al. 2023. Detecting Unknown Encrypted Malicious Traffic in Real Time via Flow Interaction Graph Analysis. In *NDSS*. ISOC.
[35] Chuanpu Fu et al. 2023. Frequency Domain Feature Based Robust Malicious Traffic Detection. *IEEE/ACM Trans. Netw.* 31, 1 (2023), 452–467.
[36] Chuanpu Fu et al. 2023. Point Cloud Analysis for ML-Based Malicious Traffic Detection: Reducing Majorities of False Positive Alarms. In *CCS*. ACM, 1005–1019.
[37] Chuanpu Fu et al. 2024. Detecting Tunneled Flooding Traffic via Deep Semantic Analysisof Packet Length Patterns. In *CCS*. ACM, 3659 – 3673.
[38] Li Gao et al. 2025. Wedjat: Detecting Sophisticated Evasion Attacks via Real-time Causal Analysis. In *KDD*. ACM, 342–353.
[39] Harm Griffioen et al. 2021. Scan, Test, Execute: Adversarial Tactics in Amplification DDoS Attacks. In *CCS*. ACM, 940–954.
[40] Dongqi Han et al. 2021. DeepAID: Interpreting and Improving Deep Learning-based Anomaly Detection in Security Applications. In *CCS*. ACM, 3197–3217.

[41] Xueying Han et al. 2024. ECNet: Robust Malicious Network Traffic Detection With Multi-View Feature and Confidence Mechanism. *IEEE Trans. Inf. Forensics Secur.* 19 (2024), 6871–6885.

[42] Zijun Hang et al. 2023. Flow-MAE: Leveraging Masked AutoEncoder for Accurate, Efficient and Robust Malicious Traffic Classification. In *RAID*. ACM, 297–314.

[43] Jordan Holland et al. 2021. New Directions in Automated Traffic Analysis. In *CCS*. ACM, 3366–3383.

[44] Arthur Selle Jacobs et al. 2022. AI/ML for Network Security: The Emperor has no Clothes. In *CCS*. ACM, 1537–1551.

[45] Syed Usman Jafri et al. 2024. Leo: Online ML-based Traffic Classification at Multi-Terabit Line Rate. In *NSDI*. USENIX.

[46] Steve T. K. Jan et al. 2020. Throwing Darts in the Dark? Detecting Bots with Limited Data using Neural Data Augmentation. In *SP*. IEEE, 1190–1206.

[47] Mobin Javed and Vern Paxson. 2013. Detecting stealthy, distributed SSH bruteforcing. In *CCS*. ACM, 85–96.

[48] Zhaoxin Jin et al. 2023. Transformer-based Model for Multi-tab Website Fingerprinting Attack. In *CCS*. ACM, 1050–1064.

[49] Mattijs Jonker et al. 2017. Millions of targets under attack: a macroscopic characterization of the DoS ecosystem. In *IMC*. ACM, 100–113.

[50] Roberto Jordaney et al. 2017. Transcend: Detecting Concept Drift in Malware Classification Models. In *Security*. USENIX, 625–642.

[51] Min Suk Kang et al. 2013. The Crossfire Attack. In *SP*. IEEE, 127–141.

[52] Min Suk Kang et al. 2016. SPIFFY: Inducing Cost-Detectability Tradeoffs for Persistent Link-Flooding Attacks. In *NDSS*. ISOC.

[53] Isaiah J. King and H. Howie Huang. 2022. Euler: Detecting Network Lateral Movement via Scalable Temporal Graph Link Prediction. In *NDSS*. ISOC.

[54] Aleksandar Kuzmanovic and Edward W. Knightly. 2003. Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants. In *SIGCOMM*. ACM, 75–86.

[55] Peiyang Li et al. 2023. Learning from Limited Heterogeneous Training Data: Meta-Learning for Unsupervised Zero-Day Web Attack Detection across Web Domains. In *CCS*. ACM, 1020–1034.

[56] Xuezixiang Li et al. 2021. PalmTree: Learning an Assembly Language Model for Instruction Embedding. In *CCS*. ACM, 3236–3251.

[57] Yuanjie Li et al. 2021. Deterrence of Intelligent DDoS via Multi-Hop Traffic Divergence. In *CCS*. ACM, 923–939.

[58] Xinjie Lin et al. 2022. ET-BERT: A Contextualized Datagram Representation with Pre-training Transformers for Encrypted Traffic Classification. In *WWW*. ACM, 633–642.

[59] Zaoxing Liu et al. 2021. Jaqen: A High-Performance Switch-Native Approach for Detecting and Mitigating Volumetric DDoS Attacks with Programmable Switches. In *Security*. USENIX, 3829–3846.

[60] Xiapu Luo and Rocky K. C. Chang. 2005. On a New Class of Pulsing Denial-of-Service Attacks and the Defense. In *NDSS*. ISOC.

[61] Zhenhao Luo et al. 2023. VulHawk: Cross-architecture Vulnerability Detection with Entropy-based Binary Code Search. In *NDSS*. ISOC.

[62] Robert Merget et al. 2019. Scalable Scanning and Automatic Classification of TLS Padding Oracle Vulnerabilities. In *Security*. USENIX, 1029–1046.

[63] Seyed Mohammad Mehdi Mirnajafizadeh et al. 2024. Enhancing Network Attack Detection with Distributed and In-Network Data Collection System. In *Security*. USENIX, 5161–5178.

[64] Yisroel Mirsky et al. 2018. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. In *NDSS*. ISOC.

[65] Mehryar Mohri et al. 2018. *Foundations of machine learning*. MIT press.

[66] Mordor. Accessed Apr. 2025. DDoS Protection Service Market Size & Share Analysis. https://www.mordorintelligence.com/industry-reports/ddos-protection-market.

[67] Sourav Panda et al. 2021. SmartWatch: accurate traffic analysis and flow-state tracking for intrusion prevention using SmartNICs. In *CoNEXT*. ACM, 60–75.

[68] Eric Pauley et al. 2025. Secure IP Address Allocation at Cloud Scale. In *NDSS*. ISOC.

[69] Julien Piet et al. 2023. Network Detection of Interactive SSH Impostors Using Deep Learning. In *Security*. USENIX, 4283–4300.

[70] Yuqi Qing et al. 2024. Low-Quality Training Data Only? A Robust Framework for Detecting Encrypted Malicious Network Traffic. In *NDSS*. ISOC.

[71] Jian Qu et al. 2023. An Input-Agnostic Hierarchical Deep Learning Framework for Traffic Fingerprinting. In *Security*. USENIX, 589–606.

[72] Christian Rossow. 2014. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *NDSS*. The Internet Society.

[73] Ravjot Singh Samra and Marinho P. Barcellos. 2023. DDoS2Vec: Flow-Level Characterisation of Volumetric DDoS Attacks at Scale. *PACMNET* 1, CoNEXT3 (2023), 13:1–13:25.

[74] Rahul Anand Sharma et al. 2022. Lumen: a framework for developing and evaluating ML-based IoT network anomaly detection. In *CoNEXT*. ACM, 59–71.

[75] Meng Shen et al. 2021. Accurate Decentralized Application Identification via Encrypted Traffic Analysis Using Graph Neural Networks. *IEEE Trans. Inf. Forensics Secur.* 16 (2021), 2367–2380.

[76] Meng Shen et al. 2023. Subverting Website Fingerprinting Defenses with Robust Traffic Representation. In *Security*. USENIX.

[77] Giuseppe Siracusano et al. 2022. Re-architecting Traffic Analysis with Neural Network Interface Cards. In *NSDI*. USENIX, 513–533.

[78] Robin Sommer and Vern Paxson. 2010. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In *SP*. IEEE, 305–316.

[79] Tushar Swamy et al. 2022. Taurus: a data plane architecture for per-packet ML. In *ASPLOS*. ACM, 1099–1114.

[80] Ruming Tang et al. 2020. ZeroWall: Detecting Zero-Day Web Attacks through Encoder-Decoder Recurrent Neural Networks. In *INFOCOM*. IEEE, 2479–2488.

[81] Florian Tegeler et al. 2012. BotFinder: finding bots in network traffic without deep packet inspection. In *CoNEXT*. ACM, 349–360.

[82] Ege Tekiner et al. 2022. A Lightweight IoT Cryptojacking Detection Mechanism in Heterogeneous Smart Home Networks. In *NDSS*. ISOC.

[83] Saravanan Thirumuruganathan et al. 2024. Detecting and Mitigating Sampling Bias in Cybersecurity with Unlabeled Data. In *Security*. USENIX, 1741–1758.

[84] Muoi Tran et al. 2019. On the Feasibility of Rerouting-Based DDoS Defenses. In *SP*. IEEE, 1169–1184.

[85] CTU University. Accessed Apr. 2025. The CTU-13 Dataset. A Labeled Dataset with Botnet, Normal and Background traffic. https://www.stratosphereips.org/datasets-ctu13.

[86] Thijs van Ede et al. 2022. DEEPCASE: Semi-Supervised Contextual Analysis of Security Events. In *SP*. IEEE, 522–539.

[87] Ashish Vaswani et al. 2017. Attention is All you Need. In *NIPS*. 5998–6008.

[88] Mathew Vermeer et al. 2023. Alert Alchemy: SOC Workflows and Decisions in the Management of NIDS Rules. In *CCS*. ACM, 2770–2784.

[89] Daniel Wagner et al. 2021. United We Stand: Collaborative Detection and Mitigation of Amplification DDoS Attacks at Scale. In *CCS*. ACM, 970–987.

[90] Kai Wang, , et al. 2023. BARS: Local Robustness Certification for Deep Learning based Traffic Analysis Systems. In *NDSS*. ISCO.

[91] Feng Wei et al. 2023. XNIDS: Explaining Deep Learning-based Network Intrusion Detection Systems for Active Intrusion Responses. In *Security*. USENIX, 4337–4354.

[92] Xi Wei et al. 2020. Multi-Modality Cross Attention Network for Image and Sentence Matching. In *CVPR*. CVF/IEEE, 10938–10947.

[93] Matthias Wichtlhuber et al. 2022. IXP scrubber: learning from blackholing traffic for ML-driven DDoS detection at scale. In *SIGCOMM*. ACM, 707–722.

[94] WIDE. Accessed Apr. 2025. MAWI Working Group Traffic Archive. http://mawi.wide.ad.jp/mawi/.

[95] Renjie Xie et al. 2023. Rosetta: Enabling Robust TLS Encrypted Traffic Classification in Diverse Network Environments with TCP-Aware Traffic Augmentation. In *Security*. USENIX, 625–642.

[96] Jiarong Xing et al. 2020. NetWarden: Mitigating Network Covert Channels while Preserving Performance. In *Security*. USENIX, 2039–2056.

[97] Jiarong Xing et al. 2021. Ripple: A Programmable, Decentralized Link-Flooding Defense Against Adaptive Adversaries. In *Security*. USENIX, 3865–3880.

[98] Songsong Xu et al. [n. d.]. One Model Fits All Nodes: Neuron Activation Pattern Analysis-Based Attack Traffic Detection Framework for P2P Networks. *IEEE/ACM Trans. Netw.* ([n. d.]), to appear.

[99] Zhiying Xu et al. 2022. Xatu: boosting existing DDoS detection systems using auxiliary signals. In *CoNEXT*. ACM, 1–17.

[100] Jinzhu Yan et al. 2024. Brain-on-Switch: Towards Advanced Intelligent Network Data Plane via NN-Driven Traffic Analysis at Line-Speed. In *NSDI*. USENIX, 419–440.

[101] Limin Yang et al. 2021. CADE: Detecting and Explaining Concept Drift Samples for Security Applications. In *Security*. USENIX, 2327–2344.

[102] Yucheng Yin et al. 2022. Practical GAN-based synthetic IP header trace generation using NetShare. In *SIGCOMM*. ACM, 458–472.

[103] Menghao Zhang et al. 2020. Poseidon: Mitigating Volumetric DDoS Attacks with Programmable Switches. In *NDSS*. ISOC.

[104] Ruijie Zhao et al. 2022. MT-FlowFormer: A Semi-Supervised Flow Transformer for Encrypted Traffic Classification. In *KDD*. ACM, 2576–2584.

[105] Ruijie Zhao et al. 2023. Yet Another Traffic Classifier: A Masked Autoencoder Based Traffic Transformer with Multi-Level Flow Representation. In *AAAI*. AAAI Press, 5420–5427.

[106] Jing Zheng et al. 2018. Realtime DDoS Defense Using COTS SDN Switches via Adaptive Correlation Analysis. *IEEE Trans. Inf. Forensics Secur.* 13, 7 (2018), 1838–1853.

[107] Guangmeng Zhou et al. 2023. NetBeacon: An Efficient Design of Intelligent Network Data Plane. In *Security*. USENIX, 6203 – 6220.

[108] Guangmeng Zhou et al. 2024. TrafficFormer: An Efficient Pre-trained Model for Traffic Data. In *SP*. IEEE, 1844–1860.

[109] Huancheng Zhou et al. 2023. Mew: Enabling Large-Scale and Dynamic Link-Flooding Defenses on Programmable Switches. In *SP*. IEEE, 3178–3192.

[110] Shitong Zhu et al. 2020. You do (not) belong here: detecting DPI evasion attacks with context learning. In *CoNEXT*. ACM, 183–197.

A Background Knowledge

A.1 Notion of Attention

As a key design feature of large language models (LLMs) [18], *attention* represents the correlation of each component relative to others. For instance, attention models how each word in a sentence relates to other words within that sentence, which is a technique widely used in natural language processing. Essentially, attention highlights correlations among elements, and the attention matrix contains the weights that represent these all-to-all correlations. We refer readers to paper [87] for details.

A.2 Computation of Attention

The self-attention model extracts correlations within one single sequence. It computes attention matrices by transforming an input matrix into query, key, and value matrices. The model then calculates attention scores by applying non-linear transformations and performing matrix multiplications with the three matrices.

Specifically, let $\mathbf{I}_{N\times L} = [\vec{l}_0, \ldots, \vec{l}_N]$ represent the input matrix, which consists of N elements denoted by vectors of length L. First, the attention model transforms the input matrix $\mathbf{I}$ into query, key, and value matrices using linear transformations:

$$\mathbf{Q} = \text{Linear}\left(\mathbf{I}; \mathbf{W}^{(Q)}, \vec{b}^{(Q)}\right) = \mathbf{I}\mathbf{W}^{(Q)} + \vec{b}^{(Q)}, \tag{23}$$

where $\mathbf{W}^{(Q)}_{L\times H}$ and $\vec{b}^{(Q)}$ are trainable parameters, i.e., the weights and biases of the linear transformation. Similarly, we can derive key and value matrices as follows:

$$\mathbf{K} = \text{Linear}\left(\mathbf{I}; \mathbf{W}^{(K)}, \vec{b}^{(K)}\right), \ \mathbf{V} = \text{Linear}\left(\mathbf{I}; \mathbf{W}^{(V)}, \vec{b}^{(V)}\right). \tag{24}$$

In the second step, the model calculates the attention matrix according to the key, value, and query matrices:

$$\mathbf{A} = \text{Atten}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathsf{T}}}{\sqrt{H}}\right)\mathbf{V}. \tag{25}$$

Here, $\mathbf{A}_{N\times N}$ denotes the attention values, where $\mathbf{A}_{i,j}$ indicates the correlation between $\vec{l}_i$ and $\vec{l}_j$ ($1 \le i, j \le N$). Furthermore, the process of computing attention is differentiable, which allows conventional training methods for deep neural networks to be used to optimize all trainable parameters.

A.3 Cross-Attention and Self-Attention

Unlike the self-attention model, which constructs $\mathbf{K}$, $\mathbf{Q}$ and $\mathbf{V}$ based on one single input, cross-attention models can analyze the correlation across multiple data sources, by constructing the key, query, and value matrices from different inputs. Our approach utilizes cross-attention to integrate features of varying granularities into a unified feature vector.

Table 5: Summary of default hyperparameter settings.

Categories	Parameters	Default	Descriptions
Feature Fusion Model	H	100	Num. hidden states.
	T	10^{-2}	Time interval.
	L	3.0	Time frame.
	N	100	Window size.
Contrastive Learning	B	50	Batch size.
	r	10^{-3}	Learning rate.

B Details of Experiment Settings

B.1 Default Hyperparameter Settings

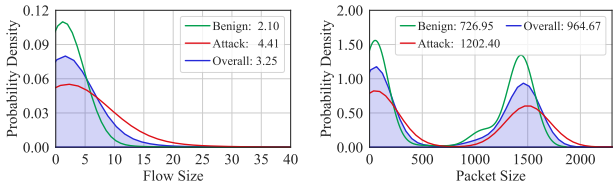
Table 5 lists the hyperparameters of tFusion. We analyze the system under various hyperparameter settings in Appendix D.4.

B.2 Details of Data Augmentation Methods

We adapt five data augmentation methods for comparison: (i) Generating traffic with fixed rules for Tor (NetAugment [6]); and for (ii) TLS traffic (Rosetta [95]); (iii) adding GAN generated abnormal features (ODDS [46]); (iv) adding traffic from manually synthesized CIC datasets [31]; and (v) Internet datasets (MAWI [94]). By default, we use these methods to compensate for 20% datasets and apply Random Forest and K-Means models to learn the flow features [28] extracted from the augmented datasets. Note that we only add benign samples from the CIC and MAWI datasets, as they do not contain the attack traffic in the testing sets. In addition, ODDS is only applicable to supervised methods, since it cannot generate benign samples.

B.3 Details of Traffic Statistics

We analyze the distribution of the traffic from the institutional network in Figure 21. Specifically, we find that, similar to traffic from the Internet backbone [94], most flows are short, consisting of 3.25 packets on average. Meanwhile, we observe that the packet sizes exhibit bimodal distributions, which is similar to the observations in existing studies [34].



(a) Flow size distribution. (b) Packet length distribution.

Figure 21: Flow size and packet size distribution.

C Details of Comparative Experiments

C.1 Comparing Different Numbers of Samples

We compare the accuracy using different numbers of samples. Table 6 shows that tFusion outperforms existing supervised and unsupervised methods with 50 training samples, by improving 17.75% and 16.34% accuracy, respectively. Moreover, the data in Figure 22 indicate that tFusion achieves 0.9863, 0.9888, 0.9914, and 0.9946 accuracy with 50, 100, 150, and 200 training samples, respectively. In contrast, the baselines can only achieve 0.7734 ~ 0.9028 AUC. Similarly, the data in Figure 23 show that unsupervised tFusion achieves 0.9419 ~ 0.9800 AUC, i.e., 12.76% ~ 19.28% improvement over existing methods.

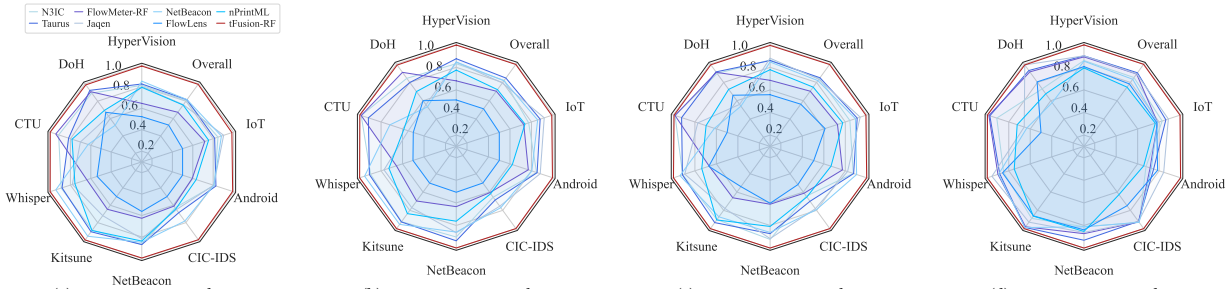
C.2 Detailed Comparison for Ablation Study

We disable the key design of the packet feature embedding module. We find that when we disable the time-aware positional encoding, i.e., only considering the sizes of packets—the overall accuracy decreases by 7.06% and 22.60% for supervised and unsupervised learning, respectively. This indicates the significance of temporal

Table 6: Detection accuracy using 50 randomly sampled flows from the datasets for training.

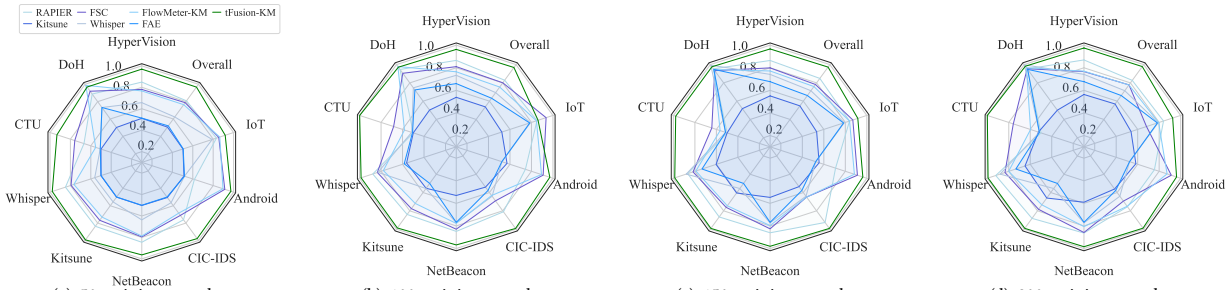
Methods[1]	HyperVision Datasets					Datasets Collected by CIC						Datasets in Exisitng Studies				Overall
	Flood	Prob	Web	Malware	Adv.	IDS'17	IDS'18	DoS'19	Android	IoT	DoH	CTU	Whisper	Kitsune	NetB.	
N3IC	0.8726	0.7555	0.7953	0.8035	0.8995	0.6208	0.6168	0.6492	0.8094	0.8473	0.8934	0.9154	0.8902	0.8863	0.8708	0.8056
Taurus	0.8815	0.7849	0.7734	0.8207	0.8948	0.7779	0.6550	0.6249	0.8271	0.8142	0.9248	0.8175	0.8821	0.9051	0.8712	0.8169
FlowMeter-RF	0.6058	0.6225	0.7021	0.6197	0.7079	0.6266	0.5948	0.6083	0.5950	0.7176	0.9090	0.9434	0.6389	0.6463	0.6257	0.6722
Jaqen	0.8176	0.6952	0.8900	0.8752	0.8350	0.7467	0.6076	0.5948	0.8049	0.9027	0.6276	0.7019	0.7813	0.7197	0.8163	0.7594
NetBeacon	0.8894	0.7240	0.8716	0.9328	0.7769	0.9398	0.8171	0.5900	0.8069	0.8806	0.5223	0.6261	0.9429	0.9571	0.8486	0.8178
FlowLens	0.5712	-	0.5496	-	0.5308	-	-	-	-	-	0.6718	-	0.5380	-	0.5623	0.5249
nPrintML	0.8812	0.7678	0.7600	0.8533	0.7102	0.5983	0.6815	0.6434	0.6336	0.7551	0.6937	0.7842	0.7558	0.8910	0.8384	0.7555
RAPIER	0.8855	0.7999	0.8705	0.8661	0.8990	0.9004	0.7964	0.5887	0.6157	0.8323	0.8872	0.6670	0.8484	0.8370	0.8451	0.8094
Kitsune	0.5020	0.5118	0.5287	0.5139	0.5092	0.5014	-	0.5144	0.5137	0.5038	0.5021	-	0.5000	0.5004	0.5021	0.5066
FSC	0.8038	0.5272	0.5701	-	0.7611	0.5271	0.6531	0.6481	0.8235	0.6340	0.9201	0.7572	0.7425	0.7677	0.7811	0.6883
Whisper	0.6589	0.8923	0.8857	0.5308	0.5647	-	-	0.6943	0.5182	0.7904	0.7325	0.5071	0.5616	0.5559	0.6365	0.6232
FlowMeter-KM	0.8038	0.5269	0.5573	-	0.7413	-	0.6478	0.6478	0.8166	0.6358	0.9875	-	0.7228	0.7311	0.7714	0.6654
FAE	-	-	0.5116	0.5308	0.5473	-	-	-	0.5182	0.5062	0.7325	0.5071	-	-	-	0.5208
tFusion-RF	0.9956	0.9962	0.9946	0.9811	0.9904	0.9972	0.9972	0.9968	0.9909	0.9834	0.9881	0.9977	0.9989	0.9997	0.9962	0.9943
tFusion-KM	0.9558	0.9413	0.9783	0.9637	0.9686	0.9823	0.9816	0.9724	0.9736	0.9180	0.9683	0.9320	0.9779	0.9936	0.9629	0.9676

[1] HyperVision [34] cannot detect attacks under this setting, because merely 50 flows cannot provide sufficient flow interaction information.



(a) 50 training samples. (b) 100 training samples. (c) 150 training samples. (d) 200 training samples.

Figure 22: tFusion enables supervised models to detect various attacks when using different numbers of samples for training.



(a) 50 training samples. (b) 100 training samples. (c) 150 training samples. (d) 200 training samples.

Figure 23: tFusion enables unsupervised models to detect various attacks when using different numbers of samples for training.

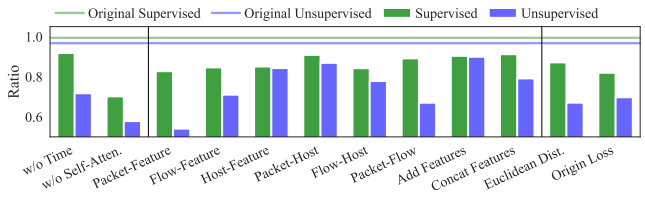
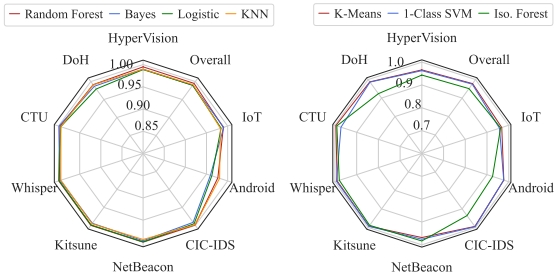


Figure 24: Ablation study with 50 training samples.

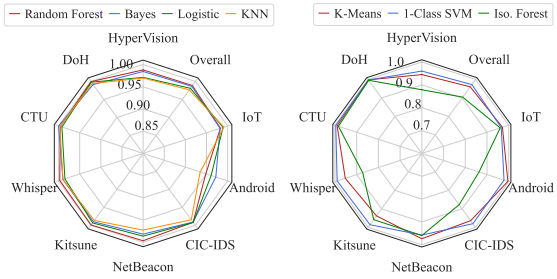
patterns for traffic detection, and implies a reason for why existing Transformer based benign traffic classification models [58, 105, 108] cannot accurately identify malicious traffic: their models cannot capture temporal patterns. In addition, we observe that the self-attention model can contribute to an accuracy increase of 28.81% to 36.94% for different ML models.

Moreover, we replace our packet embedding module with existing Transformer models for traffic classification [58, 105, 108]. These models can only achieve at most 85.5% detection accuracy attained by tFusion. Note that, without using our cross-attention module to fuse flow and host features, these Transformer models cannot achieve over 0.6 AUC accuracy. Note that these models only focus on classifying benign traffic by analyzing packet features [58, 104, 108], but cannot utilize flow and host features, which are critical for detecting attack traffic.



(a) Supervised models. (b) Unsupervised models.

Figure 25: Accuracy of ML models (50 samples).



(a) Supervised models. (b) Unsupervised models.

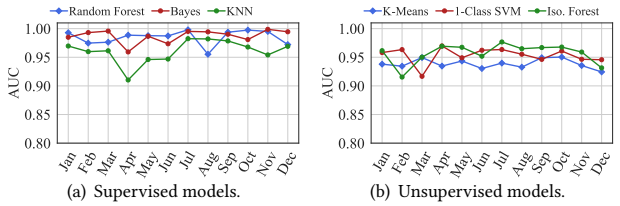
Figure 26: Accuracy of ML models (1.0 ‰ samples).

In addition, we disable the design of contrastive learning by using Euclidean distance and the existing contrastive loss function [14]. The results show that using Euclidean distance can only achieve at most 0.8715 AUC, i.e., 11.77% lower than our cosine similarity. Moreover, directly applying the contrastive loss function cannot achieve high accuracy, i.e., 0.8197 AUC, which is 16.98% lower than our loss function.

D Details of Robustness Analysis

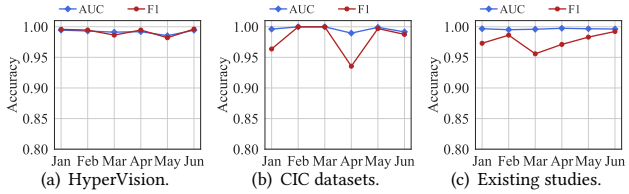
D.1 Robustness of Different ML Models

tFusion synthesizes the features via attention to provide efficient features that allow various ML models to achieve accurate detection. Currently, tFusion supports three unsupervised and four supervised models. We measure the accuracy of the models under different settings. The results are depicted in Figures 25 and 26.



(a) Supervised models. (b) Unsupervised models.

Figure 27: Accuracy of different models (Point-F 2023).



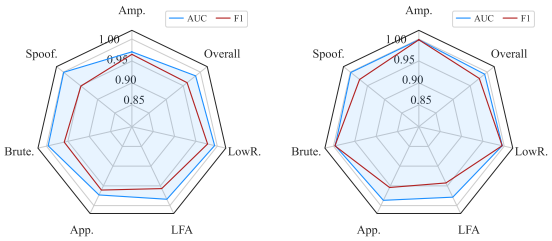
(a) HyperVision. (b) CIC datasets. (c) Existing studies.

Figure 28: Accuracy on different datasets (Point-G 2020).

We observe that, when using 50 benign and 50 attack samples to train a supervised ML model, the accuracy ranges between 0.9896 and 0.9919 AUC (see Figure 25(a)). In contrast, without using attack samples, the unsupervised models of K-Means, One-Class SVM, and Isolation Forest achieve 0.9780, 0.9754, and 0.9510 AUC, respectively. Obviously, the accuracy is lower than that of supervised models. Meanwhile, we can see that tFusion achieves high accuracy when using 1.0 ‰ datasets for training. This allows supervised and unsupervised models to achieve 0.9887 and 0.9762 AUC, respectively. Note that tFusion provides efficient features that synthesize various kinds of features, thereby enabling these lightweight models to easily capture stealthy attack traffic.

D.2 Other Potential Adversarial Attacks

Robustness Against Topology Manipulations. We consider attackers abuse encrypted tunnels to conceal topology patterns for evasion. Specifically, we replay existing tunnel traffic datasets [37], including six categories of malicious traffic delivered by IPSec tunnels along with benign traffic. From Figure 29, we observe that tFusion can identify tunneled malicious traffic and achieve 0.9861 average AUC and 0.9612 average F1. The reason is that tFusion utilizes an attention model to analyze fine-grained packet-level features, which enables it to detect attack traffic when coarse-grained topology features are obfuscated by tunnels.



(a) Unsupervised models. (b) Supervised models (50 samples).

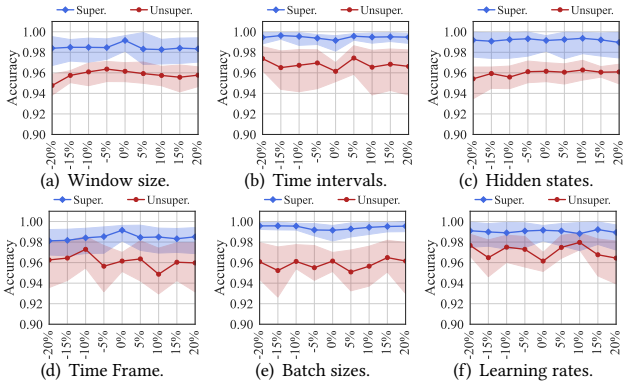
Figure 29: Accuracy in detecting tunneled malicious traffic.

Benign Traffic Precedes Malicious Traffic. Similar to the traffic obfuscation [33] discussed in Section 5.3, we send benign packet before transmitting malicious traffic, rather than injecting benign traffic at randomly selected positions. When using 50 randomly selected benign flows to train K-Means models, tFusion can achieve 0.9055 ~ 0.9765 AUC, with an average AUC of 0.9462. This represents a bounded accuracy drop of 2.51%, which is similar to the three existing evasions strategies.

D.3 Robustness w.r.t Pre-Training Datasets

tFusion requires Internet traffic datasets for pre-training. We thus analyze the robustness of performance when using different traffic datasets collected over one year. Specifically, we use traffic collected by the MAWI project from a 10 Gb/s optical fiber at Point-G. Moreover, we also use traffic from Point-F, collected from the peers of AS 2500. The results show that tFusion achieves stable performance when using different datasets for pre-training.

First, from Figure 27(a), we can see that tFusion achieves an average AUC ranging between 0.9606 and 0.9873 when providing different supervised models with traffic features extracted by models pre-trained with traffic collected over the 12 months. Similarly,



(a) Window size. (b) Time intervals. (c) Hidden states.

(d) Time Frame. (e) Batch sizes. (f) Learning rates.

Figure 30: Accuracy of different hyperparameter settings.

Figure 27(b) shows that tFusion can enable the unsupervised models to achieve AUCs ranging from 0.9385 to 0.9568 when using traffic from different months. Second, Figure 28 depicts the accuracy achieved by Random Forest models when learning features extracted by tFusion, where tFusion is pre-trained on datasets collected during the last six months before collection terminated. The results demonstrate that the accuracy fluctuations are stable across the datasets, with accuracy on different datasets ranging between 0.9805 and 0.9915 F1. Third, we compare the accuracy of tFusion when pretrained on different datasets with varying scales, i.e., traffic collected from a Gb/s-scale network (Point-F) and a 10 Gb/s network (Point-G). We observe that the impact of dataset scale can be bounded by 1.11% F1. Therefore, tFusion achieves robust detection performance regardless of the choice of pretraining datasets.

D.4 Hyperparameter Sensitivity

We increase and decrease the values of the hyperparameters by 5% to 20%. The results are shown in Figure 30. First, regarding the packet embedding module, the accuracy falls within 0.9811 to 0.9915 AUC when supervised models are selected by the AutoML module. We can also find that, when N increasing, the accuracy increases slightly. Similarly, we find that tFusion is robust against different time intervals (T), with the variance bounded by 2.42% AUC. In addition, the hyperparameters for the cross-modality fusion module, i.e., hidden states (H) and window size (L), show only a fluctuation in accuracy within 1.95%. Meanwhile, we observe an increasing trend in accuracy as the scale of the network increases (i.e., higher H). Additionally, the learning rates and batch sizes of contrastive learning do not significantly affect the accuracy, with fluctuations of -0.43% to 1.10% and -1.89% to 0.33%, respectively.