VIII. Appendix

IX. Problem Formulation

As shown in Table VI, we formulate APT detection within an active learning framework as a constrained optimization problem that jointly maximizes detection performance while minimizing labeling costs and computational overhead.

TABLE VI: Notation Definitions

Symbol	Definition
$\mathcal{X}_s$	Original statistical features
$\mathcal{X}_t$	Original sequential features
$\mathcal{X}'_s$	Preprocessed statistical features
$\mathcal{X}'_t$	Preprocessed sequential features
$\mathcal{Z}$	Concatenated feature representation
$\mathcal{Z}_{\text{pool}}$	Pool of all samples (labeled and unlabeled)
$\mathcal{Z}_{\text{large}}$	Massive unlabeled dataset for pre-training
$\mathcal{L}$	Set of labeled samples
$\mathcal{U}$	Set of unlabeled samples
$\mathcal{Q}$	Set of samples selected for labeling in an iteration
z_i	A feature vector for the i-th sample
y_i	Label for the i-th sample
$\mathcal{M}_\theta$	Detection model with parameters θ
θ_{pretrain}	Parameters after pre-training
θ_{final}	Final parameters after LoRA fine-tuning
θ_{frozen}	Frozen parameters of the pre-trained model
θ_{LoRA}	Low-rank adaptation parameters
η	Learning rate
ϵ	Convergence threshold
$\oplus$	Concatenation operation

Feature Post-processing and Data Pool. Let $\mathcal{X}_s = \{x_{s,1}, x_{s,2}, \ldots, x_{s,N}\}$ represent the statistical features and $\mathcal{X}_t = \{x_{t,1}, x_{t,2}, \ldots, x_{t,N}\}$ represent the sequential features of the encrypted traffic samples. Through preprocessing, we transform these features into $\mathcal{X}'_s = \{x'_{s,1}, x'_{s,2}, \ldots, x'_{s,N}\}$ and $\mathcal{X}'_t = \{x'_{t,1}, x'_{t,2}, \ldots, x'_{t,N}\}$ respectively. The concatenated feature representation is denoted as $\mathcal{Z} = \{z_1, z_2, \ldots, z_N\}$, where $z_i = [x'_{s,i} \oplus x'_{t,i}]$ and $\oplus$ represents the concatenation operation.

Let $\mathcal{Z}_{\text{pool}} = \{z_1, z_2, \ldots, z_N\}$, where each sample $z_i \in \mathbb{R}^d$ resides in a d-dimensional feature space. We partition this pool into two disjoint sets: the labeled set $\mathcal{L}$ and the unlabeled set $\mathcal{U}$, where $\mathcal{Z}_{\text{pool}} = \mathcal{L} \cup \mathcal{U}$ and $\mathcal{L} \cap \mathcal{U} = \emptyset$.

At a given iteration, the labeled dataset $\mathcal{L} = \{(z_1, y_1), (z_2, y_2), \ldots, (z_m, y_m)\}$ contains m samples that have been labeled by experts, where $y_i \in \{0, 1\}$ denotes the label. The unlabeled dataset $\mathcal{U} = \mathcal{Z}_{\text{pool}} \setminus \mathcal{L} = \{z_{m+1}, z_{m+2}, \ldots, z_N\}$ contains the remaining samples that have not been labeled yet.

Pre-training and Fine-tuning. Initially, we pre-train the model $\mathcal{M}$ on a large corpus of unlabeled traffic data to learn general traffic patterns and representations. This pre-training phase is formulated as:

$$\theta_{\text{pretrain}} = \arg\min_{\theta} \mathcal{L}_{\text{pretrain}}(\theta, \mathcal{Z}_{\text{large}}), \tag{23}$$

where $\mathcal{Z}_{\text{large}}$ is a massive unlabeled dataset (potentially much larger than $\mathcal{Z}_{\text{pool}}$) and $\mathcal{L}_{\text{pretrain}}$ is a self-supervised learning objective.

After that, we fine-tune the model $\mathcal{M}_\theta$ on the updated labeled traffic data $\mathcal{L}$ to specifically detect APT traffic and lightweight adaptation.The process is formulated as:

$$\theta_{\text{final}} = \arg\min_{\theta} \mathcal{L}_{\text{finetune}}(\theta, \mathcal{L}), \tag{24}$$

where $\theta = \{\theta_{\text{frozen}}, \theta_{\text{LoRA}}\}$, with θ_{frozen} being the frozen parameters of $\mathcal{M}_\theta$ and θ_{LoRA} being the low-rank adaptation parameters.

Model Prediction. The goal is to iteratively select a subset $\mathcal{Q} \subset \mathcal{U}$ for labeling, such that the detection model $\mathcal{M}_\theta$ (parameterized by θ) achieves maximum performance under a labeling budget B. Formally:

$$\mathcal{Q} = \arg \max_{\mathcal{Q} \subset \mathcal{U}, |\mathcal{Q}| \leq B} \mathbb{E}[\text{Performance}(\mathcal{M}_\theta | \mathcal{L} \cup \mathcal{Q})]. \tag{25}$$

The active learning process terminates when the labeling budget B is exhausted or the model performance converges, i.e., $\Delta\text{Performance} < \epsilon$, where ϵ is a predefined threshold.

Active Learning Iteration. The active learning process consists of the following steps:

1) Initialize the model $\mathcal{M}_\theta$ with pre-trained parameters θ_{pretrain}.
2) Train the model on the current labeled dataset $\mathcal{L}$.
3) For each sample $z_i \in \mathcal{U}$, compute the informativeness score $V(z_i)$.
4) Select the top-k samples with highest informativeness scores to form $\mathcal{Q}$.
5) Query an oracle to label samples in $\mathcal{Q}$.
6) Update $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{Q}$ and $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{Q}$.
7) Repeat steps 2-6 with ALF-guide LoRA to fine-tuning $\mathcal{M}_\theta$ until the stopping criteria are met.

During each intermediate active learning iteration t, the model parameters are updated as:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}_{\text{total}}, \tag{26}$$

where $\mathcal{L}_{\text{total}}$ is the sum of the classification loss and a regularization term.

X. Additional Results

A. Additional Results on different strategies

The additional experiment results of different active learning strategies are shown in Tables VIII-IX.

B. Additional results on trade-offs

The additional experiment results of performance and efficiency trade-offs are shown in Tables VII.

TABLE VII: Performance and efficiency trade-offs on d_2.

Base model	Fine-tuning	AL strategy	Acc(%)	F1(%)	Recall(%)	AUC	Speed	Param_To	Param_Ta	Time(s)
Pre-ALBERT	Lora	BASS	92.00	92.01	92.01	0.9935	790	11.8M	106K	10998
Pre-ALBERT	Adapter	BASS	86.21	86.21	85.79	0.9810	833	11.7M	45K	10500
Pre-ALBERT	Full	BASS	93.69	93.75	90.56	0.9967	914	11.7M	11.7M	10429
Random init	LoRA	BASS	68.53	66.77	68.21	0.9156	806	11.8M	105K	10913
Random init	Adaptor	BASS	65.39	63.58	65.39	0.9104	817	11.7M	45K	10691
Random init	Full	BASS	82.70	81.67	82.54	0.9771	935	11.7M	11.7M	10271

TABLE VIII: Comparison of active learning strategies on scenario d_1. "iters" represent iteration.

Strategy	F1(%)	iters to 80% F1	iters to F1
BASS	96.26	28	50
MAR	87.98	45	84
RAND	87.98	45	81
CONF	71.87	100	100
Diversity	87.21	49	95
Coreset	82.69	57	71
BALT	80.44	93	100
IMBL	85.64	75	98
ENT	77.65	100	100

TABLE IX: Comparison of active learning strategies on scenario d_2.

Strategy	F1(%)	iters to 80% F1	iters to F1
BASS	92.00	38	65
MAR	81.64	92	100
RAND	91.68	89	92
CONF	77.57	100	100
Diversity	90.35	68	92
Coreset	82.69	89	91
BALT	61.25	100	100
IMBL	77.65	80	95
ENT	81.64	100	100

C. Additional results on different layers

As shown in Table X, deeper networks generally improve Acc but at the cost of higher computational demands. For all configurations, the 12-layer structure achieves a favorable balance between performance and efficiency. It achieves the lowest overall F1 (96.26%) on d_2, while maintaining a reasonable prediction speed of 805 samples per second. This makes it well-suited for applications that require both high accuracy and moderate efficiency. Meanwhile, the 6-layer model performs slightly worse in terms of F1 (90.76%) but offers higher efficiency (1466 samples per second), making it a viable alternative for scenarios with stricter latency constraints or limited computational resources.

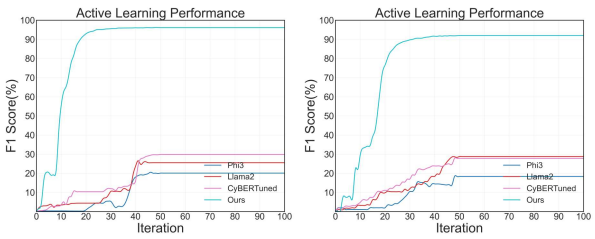
TABLE X: Compare with different layers of TrafficBASS.

Layers	Size	d_1			d_2		
		Acc(%)	F1(%)	Speed	Acc(%)	F1(%)	Speed
3	41.2M	82.93	82.83	2481	68.19	65.7	2421
6	41.74M	90.98	90.76	1466	70.74	68.28	1432
12	42.8M	96.26	96.26	805	92.01	92.01	790
24	45.98M	84.92	84.76	468	52.11	44.23	414

Although ALBERT employs parameter sharing across Transformer layers, computational complexity remains primarily influenced by forward propagation calculations. Raw traffic must traverse all Transformer layers sequentially. Computational demands increase linearly with layer count, extending processing time for deeper models. After Transformer processing, the resulting traffic representations feed into prediction components (typically fixed fully-connected layers or classifiers). These prediction components maintain consistent computational complexity regardless of model depth, resulting in similar traffic representation processing rates across models with varying layer counts.

D. Additional results on LLMs

The LLMs results are shown in Fig. 13



(a) Scenario d_1 (b) Scenario d_2

Fig. 13: Comparative F1 performance of LLMs during active learning on two scenarios.