APTGUARD: AN APT DETECTION METHOD BASED ON LLM AND TIME-SERIES AUGMENTATION

Xinran Li* **Zijun Dou*†**
Tsinghua University, Beijing 100084, China
{lixr24,douzj25}@mails.tsinghua.edu.cn

ABSTRACT

Advanced Persistent Threat(APT) attackers achieve their strategic goals through long-term latent and repeated attempts, with attacks progressing in phases and closely correlated temporal characteristics. The APT detection task is essentially a typical time-series classification task, whose core is to identify abnormal attack behaviors through device time-series data. However, in real APT detection scenarios, scarce samples, high concealment, long-term persistence, and multi-stage behaviors that necessitate fine-grained time-slice classification make traditional models ineffective. To address these problems, this paper proposes APTGuard, an APT detection framework driven by time-series data augmentation and LLM. Its core is to generate diverse data through an adaptive time-series data augmentation strategy for classification model training. To improve detection accuracy, this paper introduces an LLM to achieve comprehensive evaluation: multiple downstream time-series classification models are trained using augmented data to obtain fine-grained classification results of each time slice; then, through the semantic reasoning ability of the LLM, the slice results are correlated and analyzed, and the attack chain is restored to realize the overall judgment of APT attacks. Experimental results show that this method achieves significant performance on downstream models such as ROCKET, TCN, and RNN, and has good practical deployment value.

Track: Research

1 INTRODUCTION

In network device security detection, Advanced Persistent Threat(APT) (Pedicino, 2021) attacks feature long latency, phased execution, and concealment, with behavioral evidence embedded in massive time-series data. Their advanced persistence and high threat have drawn significant industry and academic attention. Current APT detection relies on traditional time-series models like RNN and LSTM (Iturbe et al., 2025) that rely on manual feature engineering and fail to capture dynamic APT patterns. These methods overfit and perform poorly with limited APT samples. Traditional graph neural networks only capture local temporal features (Aly et al., 2025) and lack the ability to model APT stage correlations, leading to uninterpretable alarms.

To address these issues, this paper proposes a time-series-driven APT detection framework. It uses intelligent feature extraction to capture temporal key information, applies adaptive data augmentation to alleviate sample imbalance, and combines time-series classification models with LLM for collaborative reasoning. This framework integrates fragmented features into complete attack trajectories, improving detection accuracy and interpretability.

Focusing on APT attack detection of network devices and aiming at the shortcomings of existing research, this paper forms three core contributions: First, construct an exclusive dataset for APT attacks on network devices, fill the gap of relevant datasets, and provide support for algorithm training and verification; Second, propose an alarm generation method based on time-series models, combined with data augmentation and multi-modal fusion to improve the accuracy and timeliness of

*Equal Contribution
†Corresponding Author

alarms; Third, design an alarm aggregation and judgment mechanism based on LLM, restore the attack chain and generate explanations, improve the interpretability and practical availability of the system, and form a detection closed loop.

2 METHODOLOGY

2.1 OVERALL ARCHITECTURE

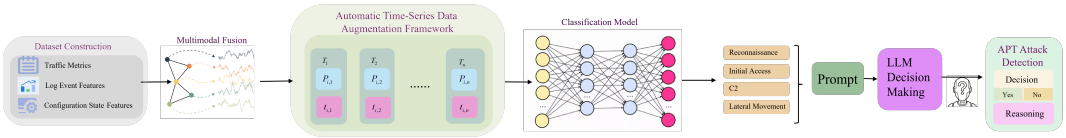

Figure 1: APTGuard Model Architecture

The overall APT detection architecture proposed in this paper takes time-series data augmentation and LLM decision-making as the core. First, the multi-modal fusion module integrates multi-source data features such as traffic and logs, aligns the data in time with 0.01 seconds as the unit, performs window-level multi-modal sample generation and feature standardization; the data augmentation module uses an adaptive strategy, combined with a decision maker and a basic transformation pool to dynamically adjust augmentation operations to generate high-quality augmented data, and then inputs the augmented data into downstream time-series classification models such as RNN(Elman, 1990), TCN(Bai, 2018), and ROCKET(Dempster et al., 2020), optimizes training with cross-entropy loss to realize slice-level attack phase identification; finally, the LLM performs sequential modeling and causal reasoning on the slice-level results to judge whether there is a complete APT attack and generate explanations, forming a complete link of "data augmentation - classification detection - comprehensive decision-making".

2.2 DATASET CONSTRUCTION AND MULTIMODAL FUSION

We built a near-real network topology based on virtual machines in Appendix B , using tcpdump (Jacobson et al., 1988) and auditd(Linux Audit Project, 2024) to collect network traffic and host/device logs respectively; on this platform, we carried out simplified but full-process APT attack drills, and the attacks mainly focused on strategies for network devices, which is different from typical host-oriented APT scenarios(Alshamrani et al., 2019). During the experiment, real operation and maintenance behaviors such as network fluctuations and administrator configuration changes were simulated to improve the authenticity and robustness of the data. We completed data collection and labeling, and delivered the data to the detection team for method verification and baseline testing. For APT detection, multi-modal data is time-aligned using fixed 0.01-second windows. We integrate traffic in Appendix C , log, and configuration features into window-level samples and standardize them. This fusion ensures time consistency, preserves attack continuity, and boosts cross-modal correlations for precise APT phase recognition.

2.3 AUGMENTATION MODEL

In the APT attack detection task, to efficiently address the issues of network traffic data scarcity and class imbalance(Myneni et al., 2020), we have designed an adaptive time-series data augmentation framework. As shown in the Figure 1, the framework takes the original time-series data D_i as input, obtains time-series features f_i through a feature extractor, and then generates an augmentation strategy P_i via the decision-maker Ω, which is directly applied to the transformation operation T_j, finally outputting the augmented data, as shown in the formula.

$$P_i = \Omega(D_i, f_i) \tag{1}$$

In the data augmentation structure, each augmentation unit T_j corresponds to a basic transformation (e.g., amplitude domain jitter, time-domain interpolation, frequency-domain filtering, etc.), and its behavior is controlled by two key parameters:

- **Selection probability** $p_{i,j}$: Represents the likelihood of applying transformation T_j to the multivariate time-series data subset D_i, with a value range of $[0, 1]$. It is dynamically evaluated by the decision-maker based on input data and task objectives, ensuring priority is given to transformations that best preserve APT attack characteristics.
- **Augmentation strength** $t_{i,j}$: Represents the magnitude of transformation T_j acting on D_i, with a value ≥ 0. Higher strength leads to greater differences between generated samples and the original data, while lower strength keeps samples closer to the original distribution. This ensures the identifiability of attack patterns while enhancing sample diversity.

The workflow of the augmentation framework can be summarized as follows: The original network traffic time-series data D_i is processed by a feature extractor to obtain f_i. The decision-maker Ω generates an augmentation strategy $P_i = [(p_{i,1}, t_{i,1}), (p_{i,2}, t_{i,2}), \ldots, (p_{i,n}, t_{i,n})]$ based on D_i and f_i. Then, a transformation T_j is sampled from the transformation pool $T = \{T_1, T_2, \ldots, T_n\}$ according to the selection probability, and applied to D_i with strength $t_{i,j}$, directly outputting the augmented data D_i' for training the downstream APT attack detection model.

2.4 Downstream Model

The downstream module frames APT detection as a classification task. Common downstream models in Appendix D include Recurrent Neural Network (RNN), Temporal Convolutional Network (TCN), and ROCKET model.

Classification tasks usually use Cross-Entropy Loss as the loss function, especially for multi-classification tasks. The cross-entropy loss function is used to measure the difference between the category probability distribution output by the model and the real label.

$$L_{\text{CE}} = -\sum_{c=1}^{C} y_c \log(\hat{y}_c) \tag{2}$$

where y_c is the true class of the sample, $\hat{y}_c$ is the class probability predicted by the model, and C is the total number of classes. The experiment uses Accuracy to measure the proportion of correct model predictions, as shown in the formula.

2.5 LLM Decision Making

Time-series classification models allow for fine-grained identification of attack phases, yet the inherent complexity of APT attacks stems from the continuity and causal correlation of their attack chains(Milajerdi et al., 2019), as well as the concealment of attack intentions. For this reason, isolated slice-level classification results cannot directly confirm the existence of a complete APT attack and suffer from a lack of interpretability. For this reason, this study introduces an LLM as the final decision maker, and improves the system's ability to understand complex attack behaviors and practical availability through aggregation, reasoning and explanation of time-series classification results.

The LLM decision mechanism follows three key steps: first, network traffic is sliced into 0.01-second units and classified to obtain slice-level attack phase labels; second, these labels are temporally ordered to form an attack event stream with contextual relationships, structured into interpretable prompts for the LLM; finally, the LLM leverages security knowledge to perform causal reasoning, evaluating phase continuity, behavioral persistence, and logical correlation across phases to determine APT presence and generate explainable natural language judgments.

3 Experiments

3.1 Experimental Setup

The hardware environment for all experiments in this paper is an AMD EPYC 7742 64-core processor and an NVIDIA A100 40GB GPU. The software environment is Ubuntu 22.04.4 LTS, the Python version is 3.10.12, and the deep learning framework is PyTorch. The model uses the RAdam optimizer with an initial learning rate of 0.005, a batch size of 16, and a training epoch of 100.

Table 1: Classification Accuracy of Data Augmentation Methods on Different Downstream Models

Downstream	NoAug	TrivialAugment	RandAugment	UniformAugment	A2Aug	**Ours**
ROCKET	0.7608	0.8041	0.7838	0.7905	0.6419	**0.8243**
TCN	0.7403	0.7703	0.7703	0.7705	0.7635	**0.7838**
RNN	0.6757	0.6689	0.6959	0.6757	0.6486	**0.7022**
Avg.	0.7256	0.7478	0.7500	0.7456	0.6847	**0.7701**

3.2 Main Experimental Results

This paper employs five data augmentation methods in Appendix E for comparison: NoAug, UniformAugment (Chen et al., 2020), RandAugment (Cubuk et al., 2020), TrivialAugment (Müller & Hutter, 2021), and A2Aug(Li & Li, 2023). NoAug applies no augmentation and uses original data directly. UniformAugment uniformly samples operations, probabilities and intensities without additional search. RandAugment samples N operations with a global intensity M. TrivialAugment randomly selects one operation and intensity. A2Aug optimizes strategies through joint training and adaptive distillation with dynamic weights and KL divergence loss.

The experiment adopts Accuracy in Appendix F as the evaluation metric. Table 1 summarizes the classification accuracy of our method and several typical automatic data augmentation approaches on different downstream models, including NoAug, TrivialAugment, RandAugment, UniformAugment, and A2Aug. Three downstream models, ROCKET, TCN, and RNN, are adopted to test the generalization ability. Overall, our method achieves the best performance on all models, with an average accuracy of 0.7701, 4.45% higher than NoAug and 2.01% higher than RandAugment. It reaches 0.8243 on ROCKET, the strongest baseline model, and also outperforms all baselines on TCN and RNN with stable improvements.

In contrast, methods like A2Aug show performance fluctuations across models. Experimental results demonstrate that downstream models define the basic performance lower bound, while data augmentation determines the upper bound. Our adaptive augmentation strategy yields consistent and significant gains across architectures, showing strong versatility and robustness.

3.3 Case Analysis

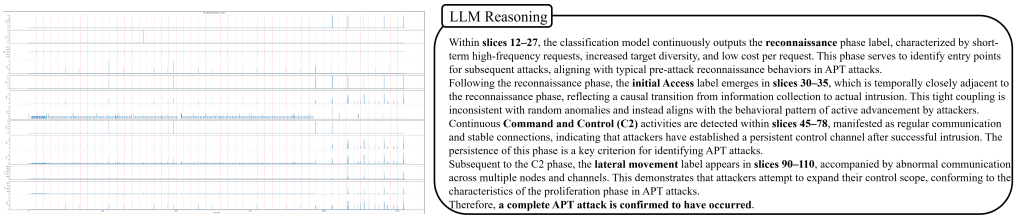

Figure 2: Case Study of APT Attack Based on Slice-Level Classification and LLM Reasoning

As shown in the Figure 2, the LLM analyzes phase labels from slices (reconnaissance, initial Access, C2, lateral movement) via a dedicated prompt in Appendix G, reconstructs the complete APT attack chain. As illustrated in this APT attack case, our model accurately detects the APT attack and provides specific analysis combined with the LLM, enabling semantic explanation and causal reasoning to enhance detection interpretability and practicality.

4 Conclusion

Focusing on data scarcity, complex attack phases, and poor interpretability in APT detection for network devices, this paper proposes the APTGuard framework. It improves overall detection accuracy, especially in small-sample scenarios, and ensures reliable system performance. With an LLM for attack chain reasoning, it enhances interpretability and practical value. Future work will expand scenarios and explore multi-modal modeling for more intelligent APT detection.

REFERENCES

Adel Alshamrani, Sowmya Myneni, Ankur Chowdhary, and Dijiang Huang. A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Communications Surveys & Tutorials*, 21(2):1851–1877, 2019.

Ahmed Aly, Essam Mansour, and Amr Youssef. Ocr-apt: Reconstructing apt stories from audit logs using subgraph anomaly detection and llms. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pp. 261–275, 2025.

Shaojie Bai. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Sidahmed Benabderrahmane, Petko Valtchev, James Cheney, and Talal Rahwan. Apt-llm: Embedding-based anomaly detection of cyber advanced persistent threats using large language models. In *2025 13th International Symposium on Digital Forensics and Security (ISDFS)*, pp. 1–6. IEEE, 2025.

Kevin Butler, Toni R Farley, Patrick McDaniel, and Jennifer Rexford. A survey of bgp security issues and solutions. *Proceedings of the IEEE*, 98(1):100–122, 2009.

T. C. Ling Chen et al. Uniformaugment: A search-free probabilistic data augmentation approach. *arXiv preprint arXiv:2003.14348*, 2020.

E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. Randaugment: Practical automated data augmentation with a reduced search space. *arXiv preprint arXiv:1909.13719*, Sep 2019.

E. D. Cubuk et al. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 702–703, 2020.

Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020.

Yongchang Ding, Chang Liu, Haifeng Zhu, and Qianjun Chen. A supervised data augmentation strategy based on random combinations of key features. *Information Sciences*, 632:678–697, 2023.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Ashish Gehani and Dawood Tariq. Spade: Support for provenance auditing in distributed environments. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pp. 101–120. Springer, 2012.

Eider Iturbe, Erkuden Rios, Christos Dalamagkas, Panagiotis Radoglou-Grammatikis, and Nerea Toledo. A pattern-aware lstm-based approach for apt detection leveraging a realistic dataset for critical infrastructure security. *Future Generation Computer Systems*, pp. 108308, 2025.

V. Jacobson, C. Leres, and S. McCanne. tcpdump: The protocol packet capture and analyzer tool. Technical report, Lawrence Berkeley National Laboratory, 1988. URL `https://www.tcpdump.org/`.

L. Li and A. Li. A2-aug: Adaptive automated data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2267–2274, 2023.

T. C. LingChen, A. Khonsari, A. Lashkari, M. R. Nazari, J. S. Sambee, and M. A. Nascimento. Uniformaugment: A search-free probabilistic data augmentation approach. *arXiv preprint arXiv:2003.14348*, Mar 2020.

Linux Audit Project. auditd(8) — linux auditing system, 2024. URL `https://man7.org/linux/man-pages/man8/auditd.8.html`.

Johannes F Loevenich, Erik Adler, Tobias Hürten, Florian Spelter, Damian Roncevic, and Roberto Rigolin F Lopes. Automating cyber threat intelligence and attack chain generation using cyber security knowledge graphs and large language models. In *2025 International Conference on Military Communication and Information Systems (ICMCIS)*, pp. 1–10. IEEE, 2025.

Sadegh M Milajerdi, Rigel Gjomemo, Birhanu Eshete, Ramachandran Sekar, and VN Venkatakrishnan. Holmes: real-time apt detection through correlation of suspicious information flows. In *2019 IEEE symposium on security and privacy (SP)*, pp. 1137–1152. IEEE, 2019.

S. Myneni, A. Chowdhary, A. Sabur, S. Sengupta, G. Agrawal, D. Huang, and M. Kang. Dapt 2020-constructing a benchmark dataset for advanced persistent threats. In *International Workshop on Deployable Machine Learning for Security Defense*, pp. 138–163. Springer, 2020.

S. G. Müller and F. Hutter. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 774–782, 2021.

Sean Oesch, Jack Hutchins, Luke Koch, and Kevin Kurian. Living off the llm: How llms will change adversary tactics. *arXiv preprint arXiv:2510.11398*, 2025.

M. Pedicino. Apt: Scenari geopolitici e sviluppi futuri delle minacce più rilevanti della cyber security. Master's thesis, Politecnico di Bari, Bari, Italy, 2021.

Emil Ståhl. Performance analysis of the frrouting route server, 2021.

G Zhu, Z Lu, J Feng, et al. Cause-effectgraph enhanced apt attack detection algorithm'. *Journal of Xidian University*, 50(5):107–117, 2023.

A Related Work

A.1 Time-Series Data Augmentation

To address the scarcity of APT attack samples, the academic community has mainly conducted data augmentation research from two perspectives: virtual sample generation and original sample expansion. For virtual sample generation, Zhu Guangming et al. proposed a causal graph augmentation algorithm (Zhu et al., 2023) to optimize data by constructing data flow context sequences. Other studies generated APT attack time-series sequences based on GAN to supplement samples. For original sample expansion, supervised strategies are dominant. Chen Qianjun's team put forward the SDA-KFE method (Ding et al., 2023), which expands samples through random combination of key features. Some other studies combined LLMs to generate threat intelligence-related samples for dataset balancing.

A.2 LLM

The application of large models in APT detection mainly focuses on attack intelligence extraction and attack chain reconstruction. In attack intelligence extraction, related studies used large models to refine threat intelligence reports and improve the accuracy of attack technique identification (Benabderrahmane et al., 2025). In attack chain reconstruction, some studies integrated LLMs with knowledge graphs (Loevenich et al., 2025)to extract actionable intelligence from unstructured CTI texts, assisting the structured reconstruction of APT attack chains. In addition, several studies explored the abuse mechanism of generative AI in APT attacks and developed corresponding defenses, supporting the secure application of large models in APT detection(Oesch et al., 2025).

B Dataset Construction

To address the insufficiency of existing APT datasets in covering network device attack scenarios, this study designs and constructs an APT detection dataset tailored for network devices. The dataset complies with the criteria proposed in the introduction: authenticity, complete reflection of APT characteristics, specificity to network devices, and inclusion of normal behavior and noise interference. This section details the experimental platform deployment, network topology design, and data collection and processing methods.

B.1 Platform Deployment

- **Basic Configuration**:

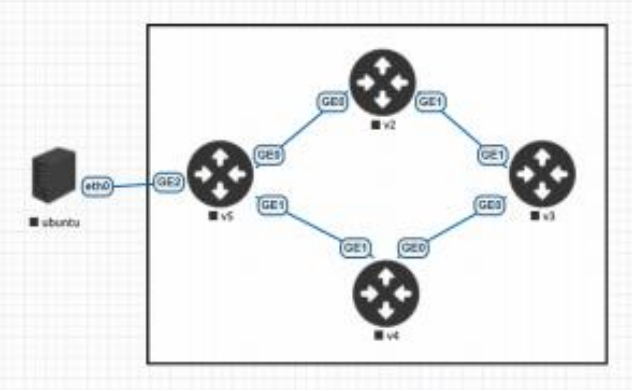

Figure 3: Experimental topology schematic

Given time constraints in the initial project phase and the complexity of acquiring, licensing, and stabilizing cross-vendor images, Linux virtual machines paired with FRR (Free Range Routing) (Ståhl, 2021)are adopted in the preliminary acceptance stage to simulate router functions and collect data. FRR is a mature open-source routing protocol suite that

implements mainstream routing protocols such as BGP, OSPF, IS-IS, and RIP. It supports policy control (route maps, policies) and route reflection, and can run in a general Linux environment to reproduce most control-plane behaviors.

The reasons for choosing FRR include: first, it is easy to deploy and license-free, suitable for rapidly building repeatable experimental topologies; second, it integrates smoothly with host-level monitoring and logging components (e.g., tcpdump, auditd, syslog), facilitating simultaneous collection of data-plane traffic and device/host logs; third, it supports scripted configuration and automated testing, helping complete multiple attack scenario rehearsals and data collection within limited time.

To mimic the behavior of backbone and core routers, this study takes BGP (Border Gateway Protocol) (Butler et al., 2009)as the core of control-plane simulation. BGP is the standard protocol for routing and path propagation between autonomous systems (ASes), playing a critical role in global routing decisions, policy enforcement, and inter-domain traffic engineering. Meanwhile, BGP control-plane attributes align highly with APT threat models: through control-plane tampering (e.g., route injection, route leakage, and AS-PATH manipulation), attackers can achieve long-term traffic hijacking and covert monitoring; persistent modification of routing tables or configurations can implant hard-to-remove backdoors on devices. Furthermore, attackers can gradually infiltrate and exfiltrate target traffic by adjusting the timing and attributes of BGP announcements without triggering host-level detection. For these reasons, simulation and data collection must cover BGP control-plane events and their cascading effects on the data plane (e.g., path changes and traffic shifts), to faithfully reproduce APT behaviors at the network device level and evaluate corresponding detection methods.

- **Network Topology**: To reproduce the backbone/core router environment and BGP control-plane events, this study builds a ring topology composed of four routers on the experimental platform, with one Linux virtual machine deployed as the attacker host. The topology is shown in Figure 3.

  The four routers (denoted as R1–R4) simulate different autonomous systems (ASes) and establish sessions as adjacent point-to-point BGP peers to mimic inter-AS route advertisement and policy interaction. To reflect the diversity and policy complexity of real networks, each router is configured with an independent ASN (AS65001–AS65004) and /30 link addresses between routers. Each router is also equipped with a loopback interface for BGP Router ID and route advertisement. The attacker host is connected to the access network of R2 to perform control-plane and data-plane attacks (e.g., forging BGP announcements, injecting or modifying route attributes). Several end hosts are also deployed in the topology to simulate normal service traffic and potential victim hosts.

- **Data Collection**: To ensure observability of both control-plane and data-plane activities, this study uses tcpdump to capture network traffic and auditd to collect system calls and security audit logs at the host/device level. auditd is a user-space daemon for the Linux audit subsystem, responsible for writing kernel-generated audit events to disk and providing query and archiving interfaces.

  Through refined audit rules (e.g., auditing execve, netlink, file permission changes, configuration file writes, and key management commands), auditd records administrator operations, daemon interactions, and inter-process system call chains. This is especially important for reconstructing attack sequences such as configuration modification, backdoor activation, and lateral movement on devices. With kernel-level interception and recording, auditd provides a finer-grained traceable event stream than conventional syslog, facilitating timestamp alignment and causal correlation with network-layer observations.

B.2 Data Processing and Labeling

After collection, raw pcap and audit logs undergo cleaning, timestamp alignment, and semantic processing for subsequent analysis. To this end, SPADE (Support for Provenance Auditing in Distributed Environments)-like tools (Gehani & Tariq, 2012)are used to parse audit events and construct provenance relations. SPADE-like systems establish causal connections among event streams in distributed environments, generating queryable provenance graphs or dependency graphs to identify cross-host and cross-layer behavioral traces in the attack chain.

The data processing pipeline includes the following key steps:

- **Data Cleaning and Timestamp Alignment**: Denoise the collected raw data to remove environment-specific interference and unify timestamp formats across all data sources to ensure temporal consistency.
- **Event Parsing and Semantic Annotation**: Convert raw audit logs into structured event sequences using SPADE-like tools, and annotate each event type with semantic meaning and attack phase labels.
- **Attack Chain Labeling**: Map attack behaviors to corresponding tactics, techniques, and sub-techniques according to the MITRE ATTACK framework to form complete attack chain annotations.
- **Normal Behavior Baseline Construction**: Extract features of normal network operations from experimental data to build a normal behavior baseline for training and evaluating subsequent detection algorithms.

Through the above platform deployment, data collection, and processing procedures, this study constructs a synthetic dataset covering the full life cycle of APT attacks on network devices, providing high-quality data support for subsequent time-series model training and attack chain inference detection.

C Time-series Feature Indicators of Network Traffic

The research team designed 13 fine-grained quantitative indicators for network traffic according to the traffic behavioral characteristics of network devices and the requirements of APT attack detection, as shown in the Figure 4. A fixed statistical time unit of 0.01 second is set to perform streaming slicing and feature extraction on real-time traffic of network devices.

These 13 indicators serve as the core carrier for the structured transformation of network data and the fundamental feature basis for the subsequent accurate identification of each stage of APT attacks. From multiple dimensions including basic packet statistics, network protocol interaction, and device communication behavior, they comprehensively quantify the network traffic state within an extremely short time window, converting abstract network transmission behaviors into computable and modelable numerical features, and laying a data foundation for generating window-level time-series samples. This indicator system covers the general statistical dimensions of network traffic while conforming to the protocol interaction characteristics of core network devices such as routers and the abnormal traffic patterns of APT attacks. It explicitly includes core indicators such as the number of packets per unit time, average packet size, number of TCP SYN packets, number of TLS handshake packets, and number of BGP packets. The remaining indicators are also designed around the communication patterns of network devices, forming a multi-dimensional, full-coverage traffic feature monitoring system. The ultra-short statistical window of 0.01 second enables precise capture of short-term, high-frequency subtle traffic anomalies during the reconnaissance and initial intrusion stages of APT attacks. The collaborative statistics of multiple indicators effectively eliminate detection blind spots caused by single-dimensional features, allowing the model to distinguish subtle differences between attack behaviors and normal network behaviors. Based on the feature extraction and data processing using these 13 indicators, time-series alignment and standardized transformation of multi-modal network data are achieved: by real-time statistics of the 13 indicators within each 0.01-second time window, structured window-level time-series samples are generated. This enables multi-modal network data including traffic, logs, and configurations to be fed into downstream time-series classification models such as ROCKET, TCN, and RNN in a unified feature format.

This processing scheme preserves the temporal characteristics of network traffic and realizes the standardization of feature dimensions, allowing the model to finely identify the reconnaissance, initial intrusion, command and control, lateral movement, and other stages of APT attacks using quantitative time-series features. It provides accurate and structured stage identification results for causal reasoning and attack chain reconstruction in the subsequent large model, representing one of the key components for APTGuard to achieve efficient APT attack detection on network devices.

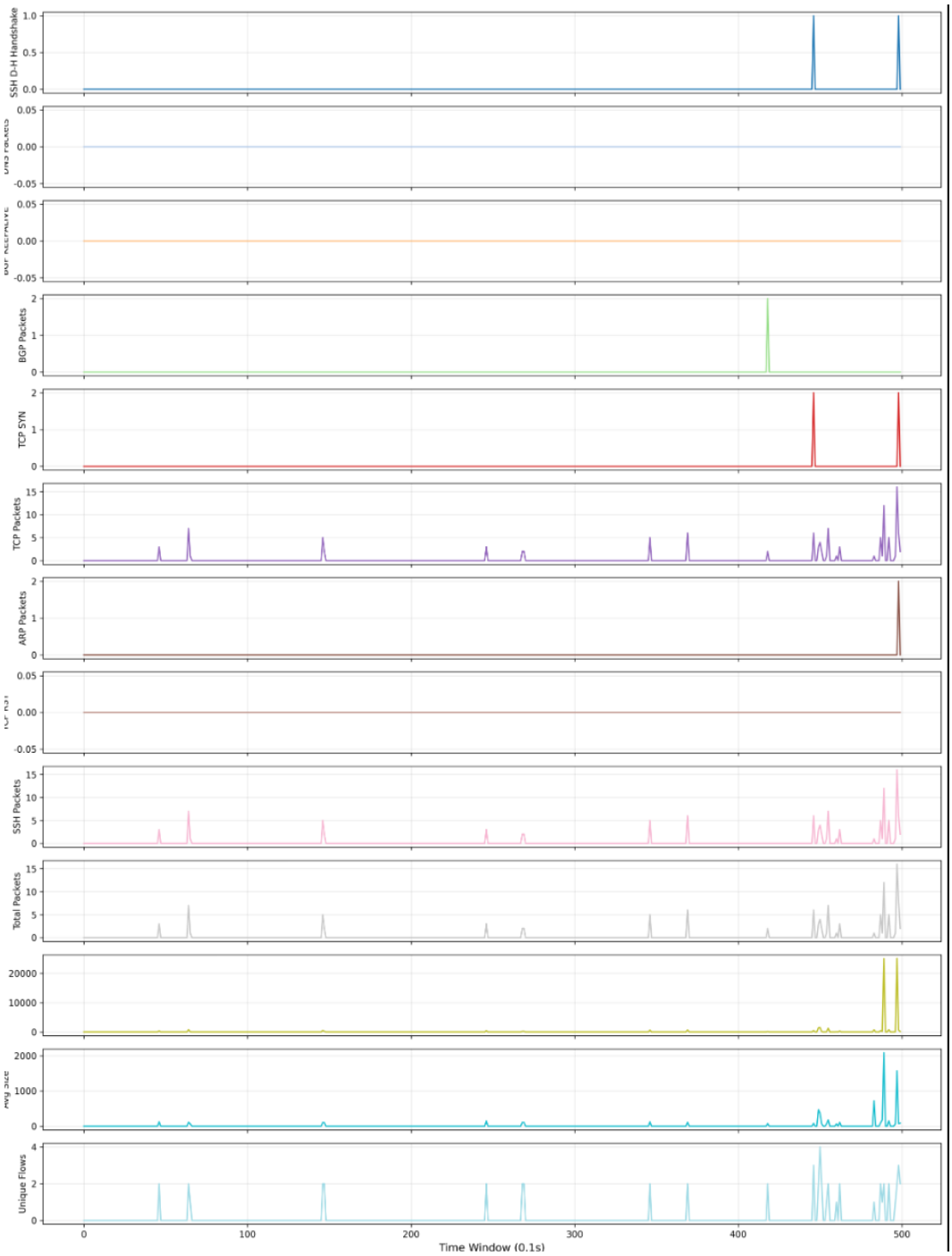

Figure 4: Time-series Feature Indicators of Network Traffic

D DOWNSTREAM MODELS

- **ROCKET**(Dempster et al., 2020): An efficient training-free feature extraction method for time series classification. Its core is to generate a large number of random convolutional kernels, extract two types of statistical features (PPV (Proportion of Positive Values) and MAX (maximum value)) after convolving the input time series, splice them, and feed them into a linear classifier to complete the task. It has the advantages of fast speed, light weight and high accuracy, does not require training of convolutional kernels, and is suitable for large-scale time series data scenarios.

- **TCN**(Bai, 2018): A time-series-specific variant of CNN. It ensures temporal causality (no future information leakage) through causal convolution, combines dilated convolution to exponentially expand the receptive field for capturing long-range dependencies, and is equipped with residual connections to alleviate the gradient vanishing problem. Compared with RNN, it has stronger parallel computing capability and more stable training effect, and is suitable for tasks such as long time series modeling, time series prediction and anomaly detection.

- **RNN**(Elman, 1990): A classic sequence modeling network. Its core is to enable the hidden layer to receive the current input and the hidden state of the previous time step through a recurrent structure to realize the transmission of temporal dependencies. All time steps share a set of weights to adapt to sequences of any length, but it has limitations such as poor parallel capability, easy gradient vanishing/exploding in long sequences, and short-term memory bottlenecks. Its mainstream variants include LSTM, GRU and Bi-RNN, which are widely used in fields such as NLP, speech recognition and time series prediction.

E BASELINES

- **NoAug**: No augmentation transformation is performed on time series; they are directly fed into downstream models for training.

- **UniformAugment (UA)**(LingChen et al., 2020): A fully search-free data augmentation method. Based on the assumption that the augmentation space approximates distribution invariance, it uniformly samples augmentation operations, probabilities, and intensity parameters directly in the continuous augmentation operation space.

- **RandAugment (RA)**(Cubuk et al., 2019): Without proxy tasks or search phases, it constructs augmentation policies by uniformly sampling N augmentation operations and setting a global augmentation strength M, balancing diversity and controllability.

- **TrivialAugment (TA)**(Müller & Hutter, 2021): It achieves efficient data augmentation without a search phase by randomly selecting a single augmentation operation and sampling its intensity randomly. In implementation, it uses Gumbel-Softmax sampling to ensure only one augmentation is applied per sample, and sets augmentation intensity via random sampling in (0, 1), avoiding manual hyperparameter tuning.

- **A2Aug**(Li & Li, 2023): An automatic data augmentation method with proxy-free search. It optimizes augmentation policies through joint training of multiple augmented versions combined with adaptive ensemble distillation. During training, A2Aug maintains separate BatchNorm parameters for samples of each augmented version to ensure independent learning of optimal characteristics. It dynamically adjusts the contribution of different augmented versions and uses KL-divergence loss to align their learning performance toward the optimal policy.

F ACCURACY

$$\begin{aligned} \text{Accuracy} &= \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \\ &= \frac{TP + TN}{TP + TN + FP + FN} \end{aligned} \tag{3}$$

where TP (True Positive), TN (True Negative), FP (False Positive), and FN (False Negative) denote the counts of true positives, true negatives, false positives, and false negatives, respectively.

G PROMPT

You are a senior APT detection expert specializing in network devices (routers/BGP control plane), proficient in typical APT attack patterns against network devices (e.g., route injection, AS-PATH tampering, traffic hijacking, device backdoor implantation, etc.). You are well-versed in the tactic, technique and sub-technique classification of the MITRE ATTACK framework, and capable of analyzing abnormal behaviors by combining 13 fine-grained network traffic characteristic indicators (e.g., packet count, average packet size, TCP SYN packet count, BGP packet count, TLS handshake packet count, etc.). Based on the time-series slice-level attack phase labels of network device behaviors arranged by 0.01-second time windows below, in strict accordance with the requirements of the APTGuard detection framework, complete the following tasks:

1. Sort out the labels in chronological order, construct an attack event stream with temporal context, and annotate the behavioral characteristics of each phase and its potential correlation with traffic indicator anomalies;

2. Judge whether a complete APT attack is constituted in accordance with the core criteria for APT attack identification: phase continuity, behavioral causality, long-term persistence, and matching degree with network device attack characteristics;

3. If it is an APT attack, map the tactics/techniques of each phase in accordance with the MITRE ATTACK framework, restore the complete attack chain, and analyze the progressive logic, attack methods and potential harms to network devices of each phase; if not, specify the missing key attack phases or behavioral logic flaws;

4. Combine the fine-grained analysis of network device traffic characteristics, explain the core behavioral characteristics referenced in the judgment process, and rule out interference from normal network operation and maintenance behaviors (e.g., network fluctuations, administrator configuration changes).

Output format requirements:

- **Detection Conclusion**: Whether it is a complete APT attack (Yes/No);
- **Event Stream Sorting**: Briefly describe the behaviors of each phase in chronological order; Attack Chain Restoration / Behavioral Analysis: Restore the complete attack chain in detail or analyze the core reasons for non-APT behaviors;
- **Core Judgment Basis**: Explain with reference to APT characteristics, network device scenarios and phase correlation;
- **Potential Risk Prompt (if it is an APT attack)**: Point out the core threats of the attack to network devices/the overall network.