Flow Microelement-Driven Traffic Relationship Analysis: Robust Detection of Malicious Encrypted Traffic

Hao Fu, Degang Sun, Jinxia Wei, Wei Wan, and Chun Long

***Abstract*—Encryption technologies randomize network communication to protect user privacy. However, attackers exploit encrypted traffic to conceal malicious activities. The existing detection methods rely primarily on traffic content or interactive patterns. Nevertheless, static methods can be easily obfuscated by advanced attacks. Since the set of potential attacks is open and infinite, models regularly lose effectiveness against novel attacks. Robust encrypted malicious traffic detection remains a valuable research area. In this paper, we propose BSTS-Net, a robust unsupervised encrypted malicious traffic detection model based entirely on traffic relations. The key motivations are to construct a relation-based traffic contextual representation and to establish dynamic baselines for anomaly detection. To represent local relations within flows, we innovatively introduce the concept of traffic microelements, which capture fine-grained interaction pattern relations. To integrate the global relationships between flows, we construct a traffic microelement space based on the Siamese neural network. Three optimization functions are proposed to optimize the intraservice, interservice and internode relations. For robust detection, we introduce a reputation-enhanced dynamic encrypted traffic detection algorithm that constructs dynamic baselines and continuously detects novel anomalies. We evaluate BSTS-Net through extensive experiments on three datasets and compare it with seven SOTA methods. Our results demonstrate its superiority, with an F1 score of more than 99.63% across all the datasets in multiclassification scenarios. Additionally, we simulate three adversarial scenarios for robustness analysis. Although the baseline methods experience an F1 score degradation of 32.21%, BSTS-Net achieves high performance, with only 1% degradation.**

***Index Terms*—Malicious traffic detection, traffic behaviour, Siamese neural network, spatial–temporal graph.**

I. Introduction

With the increasing awareness of privacy protection and security, anonymous communication technologies such as secure sockets, secure shell protocols, virtual private networks (VPNs), and Tor have been widely adopted. This trend has been followed by a significant increase in encrypted cyberattacks. Attackers exploit encrypted malicious traffic as efficient carriers, posing serious security threats with minimal costs [1]. Since cyberattacks are executed along with multi-stage network traffic, mainstream defences focus on detecting and mitigating the actual impact through traffic analysis [2]. Currently, industries widely deploy intrusion detection systems (IDSs) for malicious traffic detection. These systems (such as Suricata [3] and Snort [4]) efficiently detect encrypted traffic on the basis of threat intelligence and predefined rules [5]. However, with the emergence of novel attacks, rigid detection methods of IDSs are becoming increasingly ineffective in dealing with encrypted and obfuscated malicious traffic.

In recent years, various efforts targeting encrypted traffic detection using artificial intelligence have been proposed [6]. Most of the related research has focused on traffic content to extract complex potential attack features, and some methods have achieved accuracies exceeding 99% [7]. Some research has avoided encrypted traffic content analyses and opted for traffic interactive patterns to detect malicious communication behaviour. To leverage network communication features and enhance the representation of nodes in non-Euclidean spaces, models such as graph neural networks [8] (GNNs) and knowledge graphs [9] (KGs) integrate entities and interflow correlations. However, these static methods learn the distributional representation of malicious traffic from labelled datasets and lack robustness for novel malicious traffic. Specifically, real-life scenarios still face the following challenges:

- High-precision complex models rely on millions of high-quality labelled data for reliability, while the persistence and stealthiness of attacks complicate the labelling process [7].
- While complex model structures enhance generalization by fitting labelled datasets, the set of potential attacks is open and infinite. Detection models are updated slower than novel attacks are generated [10].
- Attackers can easily circumvent static detection methods by modifying encryption algorithms, communications protocols, etc. [11]

Anomaly-based unsupervised detection methods alleviate the aforementioned challenges to a certain extent [12]. These methods construct baselines of normal traffic and subsequently detect deviations from these baselines. Since encryption protocols generate different encrypted content for the same raw traffic content, it is impossible to find a completely closed set of benign traffic. Currently, baselines are constructed on the basis of normal traffic content or interactive behaviour,

Received 7 April 2025; revised 23 August 2025; accepted 16 September 2025. Date of publication 24 September 2025; date of current version 9 October 2025. This work was supported in part by the Youth Innovation Promotion Association of Chinese Academy of Sciences (CAS) under Grant 2022170 and in part by the Special Project for Cybersecurity and Informatization of CAS under Grant CAS-WX2022GC-04. The associate editor coordinating the review of this article and approving it for publication was Prof. Mika Ylianttila. *(Corresponding author: Chun Long.)*

The authors are with the Computer Network Information Center, University of Chinese Academy of Sciences, Beijing 100083, China (e-mail: anquanip@cnic.cn).

Digital Object Identifier 10.1109/TIFS.2025.3613971

and malicious traffic is detected by comparing reconstruction errors (e.g., Autoencoder [13]). These methods necessitate meticulous attention to the fitting of traffic content to construct distinct categorical boundaries. Since the randomness of encryption eliminates the information entropy of the content, models tend to generate false positives because of overfitting. Even though some studies integrate more universal features, such as temporal and graph structures of traffic, models trained on encrypted content are susceptible to evasion. Current anomaly-based malicious traffic detection methods fail to meet the robustness requirements necessary for reliable cybersecurity applications.

To achieve robust detection, we propose a model based on traffic microelements to extract features of relationships between similar flows. The microelement is the description of the intrinsic communication metadata pattern of given services, which remains relatively stable across multiple encrypted traffic detection instances. The model focuses on the traffic relations with better generalization rather than the encrypted content. To enhance robustness, the traffic microelements extract flow metadata as features that are less affected by encryption-related interference. While some works also propose metadata-based features, their models essentially perform static feature classification and fail to analyse correlations [14], making it difficult to dynamically detect novel attacks. Specifically, we introduce a malicious traffic detection model named BSTS-Net, which is able to exploit the fine-grained relations of traffic. By observing the mechanisms of service traffic generation, we identify their special modular fragmentation structure. On this basis, observation, a greedy strategy is employed to extract optimal traffic microelements.

Then, the traffic microelements are used to construct a traffic microelement space, which maps traffic to a suitable relational distance space. The subspace reflects high-order traffic flow behaviours. We construct datasets based on traffic microelements, including positive and negative samples, which are used to train an optimized Siamese neural network [15]. To the best of our knowledge, this is the first study in which the Siamese neural network is applied to unsupervised malicious traffic detection. To represent the exact relationships between flows, we propose three loss functions that are optimized for the intraservice, interflow, and internode flow distances. The traffic microelement space forms a similarity relationship between flows.

Finally, we construct baselines based on flows with similar relationships and obtain these baseline reputation scores. This article proposes a method for dynamic baseline construction that is based on the reputation score. We pass and update the reputation score through the temporal communication structure. Baselines with lower reputation scores will be dropped. Our approach enables dynamic, continuous and robust malicious encrypted traffic detection.

In summary, the contributions of this paper are summarized as follows:

- By analysing traffic relations, we propose a novel method for traffic microelement extraction. Intraflow and interflow relations are combined to extract optimal traffic microelements. The experiments show that the extracted traffic microelements are robust for adversarial traffic.
- We introduce the traffic microelement space, which is implemented by an optimized Siamese neural network. By incorporating traffic microelements, our method achieves efficient similarity embedding without the need for labels. We propose three targeted loss functions to improve the model performance.
- We propose a method to construct a fine-grained traffic baseline on the basis of similarity. On the basis of the baseline, we propose a dynamic reputation-enhanced malicious traffic detection model. The model analyses traffic in real time and detects unknown attacks
- We conduct experiments on three large-scale datasets, namely, CIC-IDS2017, CIC-IDS2018, and UNSWNB15, and the results demonstrate that our method outperforms the current state-of-the-art approaches. A perturbation experiment is used to verify the ability to detect adversarial attacks. Our model is complete and open source.[1]

II. Related Work

A. Content-Based Detection

Attackers hide attack intent using unique encryption algorithms, which makes traffic payloads differ. Researchers have analysed differences in traffic content distribution by building deep learning models. Wang et al. [16] first utilized CNN networks to detect the grayscale images converted from raw bidirectional traffic, which revealed the difference in traffic distribution through visualization. To improve the semantic understanding of encrypted traffic, Hwang et al. [17] proposed a novel embedding mechanism and leveraged LSTM for temporal correlation in packet headers. Recent studies have increased the model parameters for complex flow contents. Lin et al. [18] proposed a contextualized datagram representation with transformers pretrained from large-scale unlabelled data, which was applied to different encrypted traffic scenarios. Zhou et al. [19] proposed TrafficFormer, a general pretrained large model for generalization across diverse traffic analysis tasks, which achieved the best performance on six classification tasks. For robust traffic feature representation, Fu et al. [20] introduced frequency-domain analysis into the task of malicious traffic detection. Barradas et al. [21] proposed an unsupervised feature compression mechanism based on quantization and truncation, which seamlessly integrates a downstream machine learning model. Holland et al. [22] combined a unified binary packet representation, enabling end-to-end automation from raw network traffic to high-accuracy classification models. For efficient detection, Zhou et al. [23] proposed real-time traffic classification, which is implemented on a programmable switch, enabling high-speed performance. Zhao et al. [24] proposed a unified detection framework for attack traffic that supports fine-grained classification and incremental updates. In summary, content-based detection can be improved by complex modelling, while static methods are costly and unreliable for novel encrypted traffic.

B. Interaction-Based Detection

Owing to the low entropy of malicious encrypted traffic content and the difficulty of representation generalization,

[1] https://github.com/gswsfh/BSTS-Net.git

TABLE I
COMPARISON WITH RELATED WORKS

Categories	Ref.	Method Type	Feature type	Train Type	Generalization Capability	Train Costs
Traffic Content	[16]	CNN	Raw Traffic	Supervised	Low	Medium
	[17]	LSTM	Raw Traffic	Supervised	Low	Medium
	[18]	ET-BERT	Raw Traffic	Supervised	Medium	Exceptionally High
	[19]	TrafficFormer	Raw Traffic	Supervised	High	Exceptionally High
	[20]	Whisper	Metadata features	Unsupervised	Medium	Medium
	[21]	FlowLens	Metadata features	-	Low	Low
	[22]	nPrintML	Raw Traffic	Supervised	Low	Medium
	[23]	NetBeacon	Statistical Features	Supervised	Low	Low
	[24]	Trident	Statistical Features	Supervised	Low	High
Flow Interactions	[25]	RCHC	Statistical Features	Supervised	Low	Low
	[26]	KNN	Packet Length	Supervised	Low	Medium
	[27]	Semicircular DNN	Packet Length	Supervised	Medium	Medium
	[28]	GAT	Aggregation Features	Supervised	Medium	High
	[29]	RNN-evolved GCN	Aggregation Features	Supervised	Medium	High
	[30]	DLGNN	Traffic Graph	Supervised	Medium	High
	[31]	dual-domain Graph	Traffic Relationship	Supervised	Medium	High
	[32]	GAT	Traffic Graph	Supervised	Medium	High
	[33]	HyperVision	Statistical Features	Unsupervised	Medium	High
Anomaly Detection	[34]	Rosetta	Traffic Protocol	Unsupervised	Low	Medium
	[35]	Autoencoder	Statistical Features	Unsupervised	Low	Medium
	[36]	Online Learning	Traffic Packet	Self-supervised	High	High
	Ours	BSTS-Net	Microelements Relationship	Unsupervised	High	Low

researchers have aimed to detect attacks in terms of flow interactions. Gu et al. [25] discovered temporal similarities in communication within the same botnet and employed statistical algorithms to detect malicious traffic from botnets. To improve detection, researchers have used fine-grained features to represent interaction patterns. Shen et al. [26] proposed a detection method using packet length from two-way client–server interactions to train a machine learning classifier. Fu et al. [27] detected tunnelled attack traffic by analysing packet length patterns. Recent studies have constructed communication graphs for topological feature representation. Veličković et al. [28] introduced graph attention networks (GATs), which enhance node features using message passing methods. Pareja et al. [29] proposed a model that dynamically adjusts graph convolutional neural networks to capture the dynamics of graph sequence data. To incorporate temporal features, Duan et al. [30] presented sequences of spatiotemporal graphs and analysed their evolutionary relationships. To better integrate various graph interactions, Li et al. [31] transformed user features and structural information into heterogeneous graphs on the basis of relationship weights. Fu et al. [33] constructed lightweight flow interaction graphs to capture multiflow collaboration patterns in fully encrypted environments. Although interaction-based approaches have improved detection, graph models are sensitive to perturbed network environments. Consequently, [32] optimized the graph structure for noise, but the models still lack robustness to address novel malicious traffic interactions.

Graph-based methods struggle with dynamic environments because of poor generalization, adversarial fragility, and high computational cost. GNNs require global graph traversal, causing high latency in incremental updates. Their memory overhead scales linearly with graph size, limiting scalability. Attackers can manipulate topology via malicious nodes or edges, distorting neighbourhood relationships and misleading predictions. These issues hinder real-time, large-scale, robust deployment in evolving network settings.

C. Anomaly-Based Detection

To address the problem of weak robustness of malicious traffic detection models trained on limited malicious type datasets, some studies have used anomaly-based detection methods. These methods use self-supervised or unsupervised approaches to construct normal traffic baselines and detect deviation anomalies. Xie et al. [34] proposed Rosetta to accurately construct a baseline for TLS-encrypted traffic, enhancing classification performance across various network environments. Kitsune [35] introduced an autoencoder to learn from normal traffic and detect abnormal network attacks for various network channel patterns. The existing anomaly-based detection methods rely on traffic content or interactions for baseline construction, which establish classification boundaries by analysing the closure of traffic behaviour. However, evolving networks lead to high false-positive rates, which is particularly common in large-scale networks. Nakip and Gelenbe [36] proposed an online learning detection method without human involvement to automatically adapt models to novel malicious traffic. However, long-term baselines produce biases in classification boundaries, resulting in unreliable results. Moreover, detection models exhibit significant hysteresis in response to novel malicious traffic.

We improve anomaly-based detectors by exploiting flow relationships and computing local baselines on the fly, eliminating long-term bias. Compared with existing anomaly-based methods, the present method demonstrates superior robustness. Table I summarizes the relevant works.

III. MOTIVATION

Advances in traffic encryption have resulted in increasingly unreliable content-based analysis methods since traffic is encrypted into entirely unrelated random payloads at different levels of access. As highlighted by Papadogiannaki and Ioannidis [37] in their survey, traditional malware traffic detection methods that rely on payload-based features have become significantly less effective because of the widespread adoption of encryption technologies such as TLS 1.3 and encrypted server name indication. To investigate the information content of encrypted traffic payloads, we design a dedicated experiment aimed at assessing whether communication patterns generate distinguishable and meaningful payload-level features despite encryption. Specifically, we establish a proxy server using Nginx to access static resources and utilize Scapy to

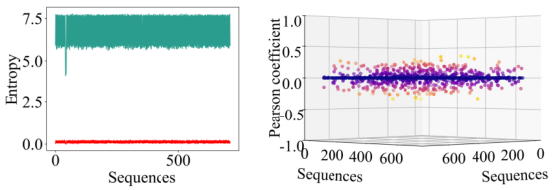


Fig. 1. Packet information entropy and the difference between and maximum randomness.

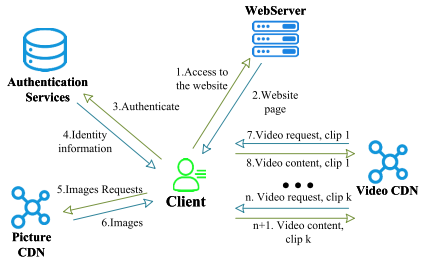


Fig. 2. Traffic componentized communication sequence diagram.

capture the encrypted packets. Fig. 1 shows the information entropy and Pearson coefficient of the encrypted packet during 1000 repeated accesses. The experimental results demonstrate several key characteristics of the encrypted packets:

- They exhibit a high level of information entropy, with values exceeding 5.7. The difference from the maximum random information entropy is less than 0.27.
- The absolute value of the Pearson coefficient between encrypted packets that originate from the same data remains within 0.25, indicating nearly uncorrelated relations.
- Overall, the entropies of the encrypted packets appear to be highly random and exhibit less significant differentiation.

Although the traffic payload is entirely indivisible, the traffic interactions exhibit a modular pattern [38], [39], [40], revealing the behaviour of traffic relations. As illustrated in Fig. 2, users access a video service through multiple requests. Initially, users visit a website and authenticate their identify through requests and then access the images and video data through staged media requests. Multiple accesses follow a similar interactive mode, and segmented communication reveals the identities.

For further research, we analyse the access patterns during the communication process in the same encrypted traffic analysis experiments. Fig. 3 illustrates the sequence of packet lengths and time intervals during the 1000 accesses. Fig. 3 (a) and (b) illustrate the distribution of the packet lengths and sequential lengths, respectively. Fig. 3 (c) and (d) illustrate the distribution of the time intervals and sequential time intervals, respectively. The results indicate that the packet lengths and intervals of the same service are confined within finite sets and exhibit stable and consistent sequential characteristics.

Combining the results of the theoretical analysis and experimental verification, we obtain the following conclusions:

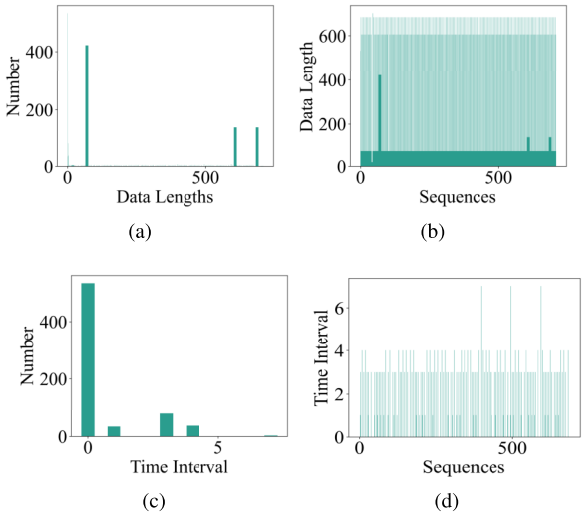


Fig. 3. Packet length and time series statistical analysis.

- Traffic data that were once homogeneous are now segmented because of the finer-grained divisions [41]. These metadata leads to more traffic information. Similar services utilizing common modules generate comparable fragments [42]. As illustrated in Fig. 2, the same video services produce analogous traffic segments when accessed by different users, and different video services also produce partly analogous traffic segments. Although the content of the encrypted traffic has become completely indistinguishable, its traffic metadata information has a large degree of similarity. Although there are variations in packets due to the content distinction, common frames and static resources result in common patterns.
- The change in design pattern introduces temporal features [43]. As illustrated in Fig. 2, the authenticated flows precede the resource flows. Flows from the same service exhibit consistent access temporal patterns. Different packets from flows also possess temporal features (e.g., heartbeat packets, etc.). The temporal features are crucial for detecting malicious traffic. On the one hand, traffic generated by scripts and tools exhibits strong periodicity. In contrast, attackers use high-frequency attack flows to improve intrusion efficiency (e.g., probe scanning) or directly cause damage (e.g., DDoS attacks). Thus, temporal features reflect traffic intentions to some extent.
- Flows of similar services result in more granular similarities [44]. The communication patterns of normal service traffic are highly similar and prevalent, forming a majority that differs sharply from the distinct and minority malicious traffic. Moreover, the finer-grained design pattern results in more obvious service behaviours [14] between benign and malicious.

Inspired by the scientific findings above, sophisticated design methodologies are needed. Specifically, we need to construct the following core functional components:

- Extraction of traffic relations from network flows.
- Construction of dynamic relevance based on traffic relations.
- Detection of malicious traffic through deviation analysis from established relevance patterns.

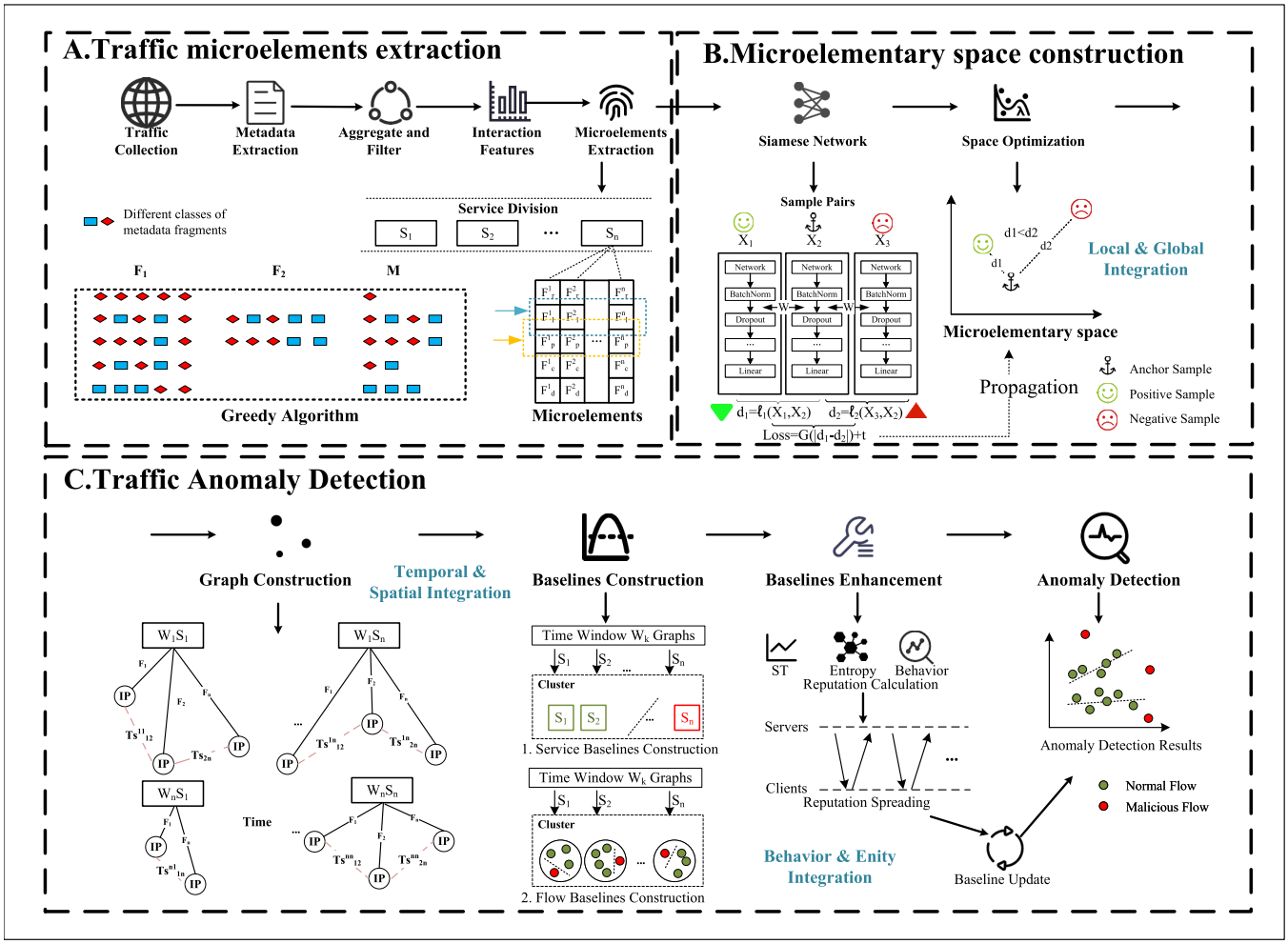


Fig. 4. System architecture diagram.

We build the above components in steps. First, we analyse the traffic metadata within services and identify the most representative as traffic microelements, which belong to the most common interactive patterns between traffic and capture the essential features of services. Second, we construct a traffic microelement space that uses an optimized Siamese neural network to determine the relevance between flows. Third, we construct a dynamic baseline for aggregated traffic on the basis of time windows and assign reputation. Malicious encrypted traffic is identified with dynamic baselines. In fact, our detection integrates various perspectives. By integrating behavioural patterns and entities, we construct traffic microelements to achieve dynamic optimality. By integrating local and global features, we construct integrated distance representations between flows. By integrating temporal and spatial dimensions, we achieve robust malicious detection based on dynamic local aggregation. In Section IV, we elaborate on the methodology employed to address the identified issues.

IV. Model Structure

To address the challenge of robust malicious traffic detection, we introduce a dynamic detection method that leverages traffic relations named BSTS-Net. Fig. 4 illustrates the overall structure of BSTS-Net, which consists of three main components–traffic microelement extraction, the SNN (Siamese neural network), and anomaly detection.

TABLE II
Symbol Meanings

Symbols	Meanings
S, SN, SF	Services, collection services and full service
F_r, F_l, F_p, F_c, F_d	Repetition, local, packet, cyclical, duplicate microelements
τ_1, τ_2, τ_3	Intra-service,inter-service,inter-node loss
R_b, R_n	Baseline and Client reputation

A. Notation

For clarity, we summarize the key notations used throughout this paper. All the symbols are defined at their first appearance, and this table II serves as a consolidated reference.

B. Traffic Microelement Extraction

The proposed microelements are designed to derive stable metadata fingerprints from multiple flows F communicating with S. The fingerprint, composed of packet slices and their sequential patterns, captures consistent and recurring communication behaviours while ignoring user-specific payload variations.

TABLE III
FEATURE SELECTION FOR FILTERING NOISY SAMPLE DATA

Features	Category
Flow number of services	numerical
IP number of services	numerical
Port number of services	numerical

A novel end-to-end approach for extracting traffic features is introduced by identifying traffic microelements, which search for as many similar interactions as possible from the traffic of the same service. The method encompasses three steps: Zeek-based feature extraction, flow feature aggregation and data filtering, and a greedy traffic microelement extraction process.

1) Zeek Feature Extraction: To obtain a more robust and generalized representation of flow interactions, we extract metadata from raw traffic using Zeek [45]. We avoid using raw metadata features directly. Instead, we leverage these metadata to construct microelements, which capture higher-level patterns for relation understanding. Since Zeek efficiently handles packet reordering, loss, and corruption during transmission [46], it also enables direct alerts on low-level link errors. Specifically, we develop a custom Zeek script,[2] spanning over 1000 lines for various common protocols, to extract the following information:

- 4-tuple information, including source IP, destination IP, source port, and destination port.
- Packet timestamp and payload size.
- Application layer metadata of traffic. This information extracted by Zeek verifies the effectiveness of traffic microelements.

2) Flow Feature Aggregation and Data Filtering: As mentioned earlier, we extract traffic microelements under services provided by specific servers [47]. Therefore, we first aggregate flow metadata based on 4-tuples, and flows under the same destination IP are considered the same service traffic. Afterwards, the following processes are carried out on each flow to extract fine-grained interactions:

- Record the direction of the packet.
- Sort the packets with metadata on the basis of the timestamp.

In fact, services such as CDN [48] generate a large number of accesses infrequently, which produces a significant amount of diverse noise traffic. We filter the noise that is irrelevant for the traffic microelements before detection. Specifically, the DBSCAN algorithm [49] is used for clustering, and the features extracted are shown in Table III. On the one hand, DBSCAN enables fast detection by leveraging precomputed distance metrics. On the other hand, it performs coarse-grained grouping, focusing only on the salient and distinctive characteristics of CDN traffic to ensure stable and efficient batch filtering. After that, we filter out the most isolated IPs whose traffic can be regarded as noise. Experiments on the CIC-IDS2017 dataset [50] demonstrate that the filtering reduces traffic by approximately 17%, without introducing any false positives. In the analysis, the IPs filtered include DNS servers and benign websites.

[2]https://github.com/gswsfh/BSTS-Net/blob/main/AllFeas.zeek

3) Greedy Traffic Microelement Extraction: For service S, we model its intrinsic behaviour from observed traffic $S_N = \{F_{N_1}, F_{N_2}, \ldots, F_{N_n}\}$, which consists of fragmented flows induced by heterogeneous user access patterns and each corresponding to distinct phases such as authentication or data transfer. By grouping flows on the basis of service, we extract microelements via the longest matching patterns, capturing a stable behavioural fingerprint that abstracts transient details while preserving invariant protocol-level interactions. During training, microfeatures are derived from a clean and comprehensive dataset S_F, enabling optimal baseline construction. In real-time detection, where full traffic S_F is unavailable, a greedy on-the-fly extraction algorithm ensures low latency and achieves near-optimal matching on the basis of $S_N \subseteq S_F$. A reliable fingerprint requires at least two flows per functional group for comparison, with statistical stability achieved at three or more flows. The success probability P of fingerprint generation is defined in Eq. 1

$$P = \left(N_t \geq 2N_g\right) \cdot \frac{T(N_t, N_g)}{S(N_t, N_g)} \tag{1}$$

where N_t is the number of traffic fragments and N_g is the number of functional groups. Here, $S(N_t, N_g)$ denotes the Stirling numbers of the second kind, representing the number of ways to partition N_t distinguishable elements into N_g nonempty unordered subsets. $T(N_t, N_g)$ represents the number of such partitions where each subset contains at least two elements, ensuring meaningful group-level matching. We compute $S(n, m)$ using the recurrence relation as shown in Eq. 2.

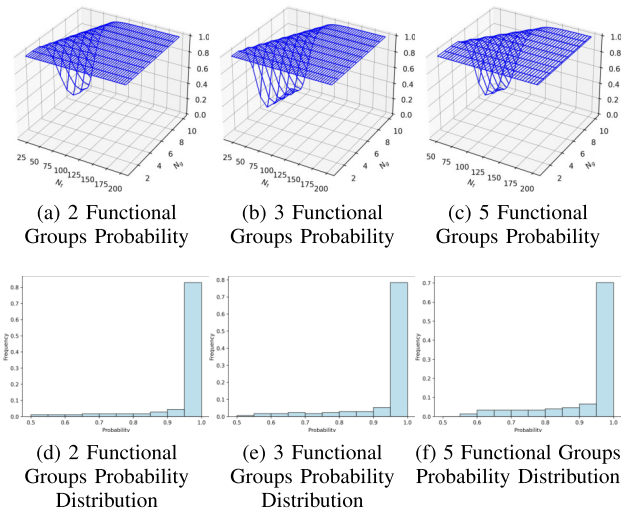
$$\begin{aligned} &S(n,m) = m \cdot S(n-1,m) + S(n-1,m-1) \\ &S(0,0) = 1, S(n,1) = 1, S(n,n) = 1 \\ &S(n,0) = 0, for\, n > 0, S(0,m) = 0, for\, m > 0 \end{aligned} \tag{2}$$

Similarly, $T(n, m)$ is computed via Eq. 3

$$\begin{aligned} &T(n,m) = m \cdot T(n-1,m) + \binom{n-1}{1} T(n-2,m-1) \\ &T(0,0) = 1, T(2m,m) = \frac{(2m)!}{m! \cdot 2^m} \\ &T(n,0) = 0, for\, n > 0, T(0,m) = 0, for\, m > 0 \\ &T(n,m) = 0, if\, n < 2m \end{aligned} \tag{3}$$

We evaluate the probability over up to 200 traffic fragments and 10 functional groups. As shown in Fig. 5, our greedy algorithm achieves a 96% success rate in generating valid traffic fingerprints, with 94.92% of matches being stable and 93.34% for groups of size 5. The histogram further shows that when there are two functional groups, more than 80% of the configurations achieve a matching probability above 95%, which holds for more than 75% of the cases with three groups and 70% of the cases with five groups, demonstrating the robustness of our approach under practical grouping scenarios.

The analysis of the CICIDS2017 dataset reveals that 86.74% of services exhibit 3 or fewer functional groups, and 95.87% have 5 or fewer functional groups. Under these realistic conditions, our method achieves a matching rate of 99.78%, demonstrating the high effectiveness and practicality of the proposed greedy microelement extraction strategy for real-world deployment.



(a) 2 Functional Groups Probability (b) 3 Functional Groups Probability (c) 5 Functional Groups Probability

(d) 2 Functional Groups Probability Distribution (e) 3 Functional Groups Probability Distribution (f) 5 Functional Groups Probability Distribution

Fig. 5. Fingerprint matching probability.

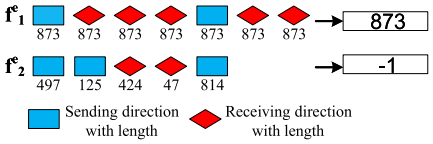


Fig. 6. Example for $\vec{F_f^r}$.

Within the traffic of the same service, we identify and extract a wealth of traffic interaction patterns. Specifically, we focus on the following five microelements:

- Repetition microelements. Protocols' (e.g., SMB and MDNS) fixed packet sizes [51], [52] allows for the categorization of packets with fixed patterns. Specifically, for flow $f = \{p_s^1, p_s^2, \ldots, p_s^k\}$, the repeatability feature is computed as Eq. (4)

$$\vec{F_f^\tau} = \mathbf{1}_{\{\|f\|>\theta_r \wedge \|\{f\}\|=1\}} \cdot (p_s^1 + 1) - 1 \tag{4}$$

where$\|\{f\}\|$ indicates the type of packets in the flow and $\|f\|$ indicates the number of packets in the flow. As illustrated in Figure 6, flow pairs with identical packet lengths are assigned feature values of 873. For flows that do not meet this criterion, the default feature value is set to -1.
- Local microelements. Owing to the similarity of request or response fragments within the flow under the same service, we employ a greedy algorithm to extract the most common fragments between traffic. Specifically, we compare the current traffic with historical traffic, identify similar fragments, and flag the longest sequence of matches. The local feature $\vec{F_f^l}$ is computed as Eq. (5)

$$\underset{f_p \sim f_s}{\operatorname{argmax}} \underset{i\in[0,l_{min}]}{\operatorname{argmin}} \zeta^i(\chi(f_c), \chi(f_p)), \vec{F_f^l} = \zeta(f_c, f_p)\| \tag{5}$$

where f_c is the current flow to be matched, f_p is the matched flow, and f_s is the flow of service. $\chi(\bullet)$ denotes the split function, which splits flows by direction. $\zeta(\bullet)$ finds common segments of flows. $\|$ combines all segments and l_{min} denotes the minimum length of the flows.

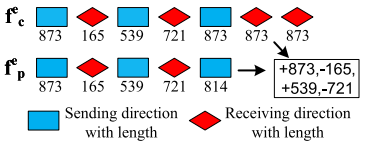


Fig. 7. Example for $\vec{F_f^l}$.

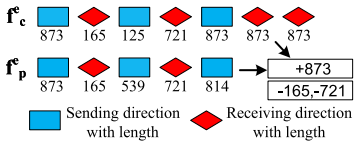


Fig. 8. Example for $\vec{F_f^p}$.

Owing to the complexity of network environments, fuzzy matching based on the forwards and backwards positions is used to address traffic fragmentation and manage packet backlog to achieve robust feature extraction. Matching segments are considered successful when they satisfy the following rules:

- The size of the segment matches the size of the target segment.
- The size of the segment matches the size of the target segment's preceding or succeeding segment in the same direction.
- The size of the segment matches the sum of the size of the target segment's preceding and succeeding segment in the same direction.

The specific process is shown in Algorithm 1.
As illustrated in Figure 7, the longest local packet sequence common to the same service is matched as 873, 165, 539, and 721. The positive and negative signs associated with these values indicate the direction of each packet.

- Packet microelements. Similarly, we implement the greedy algorithm to identify bidirectional traffic segments that exhibit similar features within the same service. The packet feature is computed by Eq.(6)

$$\vec{F_f^p} = \underset{f_p \sim f_s}{\operatorname{argmax}} \zeta(f_{c_t}, f_{p_t})\| \underset{f_p \sim f_s}{\operatorname{argmax}} \zeta(f_{c_f}, f_{p_f}) \tag{6}$$

where $\|$ represents the splice of vectors. Unlike local feature extraction, this algorithm employs a direction-based correlation, which resolves the issue of long-distance data dependency. The matching rules are as follows:

- When the current direction traffic and the target traffic have a common packet size, calculate the minimum match quantity, and save the matching segment.
- Choose the segment with the largest total matching quantity as the best matching segment. The specific algorithm is shown in Algorithm 2.

As shown in Figure 8, forwards and reverse packet sequences are matched separately, producing directional matching patterns. The forwards direction produces the sequence 873, whereas the reverse direction yields 165 and 721. Each packet is matched only once to ensure an

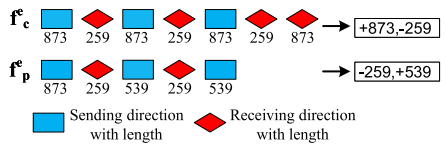


Fig. 9. Example for $\vec{F^c_f}$.

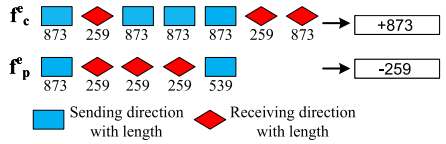


Fig. 10. Example for $\vec{F^d_f}$.

accurate representation of the communication volume and prevent duplication.

- Cyclical microelements. As illustrated in Fig. 2, the video service periodically fetches segments of resources after the initial request. Services generate periodic data packets. Accordingly, we compute the directional periodic segments of flows and count the repetitions. The circle feature is computed as Eq. (7)

$$\vec{F^c_f} = \underset{i\in[0,l_{min}]}{\text{argmax}}\, \Xi^i(\chi(f_c)) \tag{7}$$

where $\Xi(\bullet)$ calculates the number of repetitions. Common protocols, such as HTTPS [53], generate data packets of size 0 no less than twice and interference with features. Therefore, segments of size 0 need to be repeated at least 3 times. Figure 9 shows successful matching of repeated traffic segments across flows. The first segment contains 853 and 259 packets, and the second segment contains 259 and 539 packets. Tye packet signs indicate the direction, with positive signs for outbound traffic and negative signs for inbound traffic.

Algorithm 1 Local Feature Extraction

Input: Flow metadata F;
Output: Local microelements **LF**;

```
Segmenting the traffic metadata by service yields G
R = newHashTable()
for F_c, F_m ∈ G do
store S_m = GetLocalF(F_c, F_m) if R[F_c, F_m] is not stored.
LF(F_c) = the longest of all S_m
end for
return LF
```

- Duplicate features. Services generate a large number of duplicate message fragments. For example, proxy services transport data with the same size packets. In this paper, we extract the duplicated segment size within a flow and choose segments whose number of duplicates is no less than 2. The duplicate feature is computed as Eq. (8).

$$\vec{F^d_f} = \underset{i\in[0,l_{min}]}{\text{argmax}}(\Xi(f^i_c)) \tag{8}$$

Figure 10 shows the matched periodic patterns from two flows, identified as 873 and 259. Tye packet signs indicate the direction, with positive signs for outbound transmissions and negative signs for inbound transmissions.

Algorithm 2 Packet Feature Extraction

Input: Flow metadata: F;
Output: Packet microelements:**PF**;

```
Segmenting the traffic metadata by service yields G,N_b = −1,S_b
R = newHashTable()
for f_d ∈ [True, False] do
for F_c, F_m ∈ G do
store S_m = GetDirFinger(F_c[f_d], F_m[f_d]) if R[F_c, F_m] is not stored.
PF(F_c, F_m)[f_d] = the longest of all S_m
end for
end for
return PF
```

Additionally, we do not utilize a transport layer or other headers of traffic, since this information, such as the window sizes, is mostly influenced by network conditions. Even so, we can still efficiently detect protocol-targeted attacks such as slow DDoS using Zeek. Compared with the other potential features, the uniqueness of these five features is as follows:

- Among all traffic metadata, these microelements are derived from the packet length and direction, which remain highly stable across diverse encrypted traffic and can be extracted efficiently to enable real-time detection.
- These microelements are comprehensive, covering all the metadata features used in the existing studies, except for the time metrics and header fields, which are susceptible to networks. For example, they cover all nontemporal features among the more than 80 attributes extracted by CICFlowMeter [54], a tool widely used for malicious encrypted traffic detection.
- These microelements are both independent and indispensable. Empirical analysis of the CICIDS2017 dataset has demonstrated that pairwise mutual information is less than 0.15 and that additional features contribute less than 0.5% to the F1 score, satisfying the conditions for minimally sufficient statistics.

C. Microelementary Space Construction

Traffic microelements cannot be directly used as features for detection. Since different traffic microelements exhibit significant variability, making it challenging to achieve a consistent and reasonable vector representation in Euclidean space. To address this, we employ a feature mapping method based on a Siamese neural network. The Siamese neural network (SNN) [15] is a special type of neural network architecture that is typically used to process paired data. It consists of two (or more) weight-sharing subnetworks that independently process the input data and then compare their respective outputs. Owing to weight sharing, the SNN can learn more general and robust feature representations during training and improve feature extraction.

We adopt an SNN in place of AE-based approaches such as variational autoencoders (VAEs) or graph autoencoders (GAEs), which focus on reconstruction and lack explicit

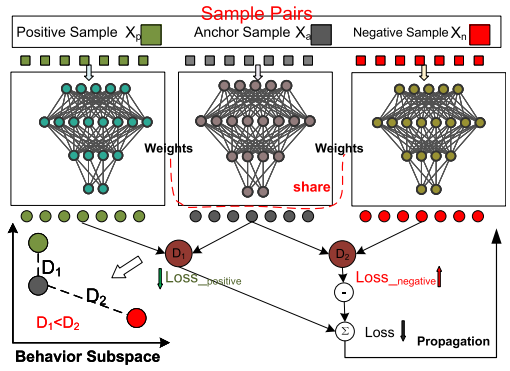


Fig. 11. Siamese network structure.

modelling of pairwise similarity. In contrast, the SNN is specifically designed to learn a similarity-preserving embedding, where instance distances directly reflect behavioural relevance.

Algorithm 3 Sampling Method

Input: Fingerprints group: F_g, sample size:N_s;
Output: Datasets:**D**;

```
for all F_g^s ∈ F_g do
    for g_c, g_p ∈ F_g^s do
        if len(g_c) < 2 then
            continue
        end if
        for i = 0, 1, ··· N_s do
            S_an, S_ac ← random(g_c), S_ne ← random(g_p)
            D.add([S_an, S_ac, S_ne])
        end for
    end for
end for
return D
```

Dataset Construct Beforehand, we propose a method to construct positive and negative sample pairs for the model, which is named the microelement dataset. As mentioned previously, traffic microelements indicate traffic relationships, and the pairs are divided on the basis of microelements. We choose a flow from the service as the anchor data and select positive data that share two or more microelements; the negative data are selected randomly from the rest of the service. The specific sampling method is illustrated in Algorithm 3.

Model Structure Subsequently, we train the SNN model. As illustrated in Fig. 11, the Siamese neural network uses different networks for positive and negative sample pairs while sharing model weights. Specifically, we employ fully connected layers, along with batch normalization and dropout, to prevent overfitting.

The loss function minimizes the distance between outputs of similar samples and maximizes the distance between the outputs of dissimilar samples. The intraservice loss is represented as Eq. (9)

$$\tau_1 = \max\{\|S_{an}, S_{ac}\| - \|S_{an}, S_{ne}\| + \epsilon, 0\}$$
$$\|S_x, S_y\| = \sum_{i=0}^{l_{max}} S_x^i - S_y^i \tag{9}$$

where ϵ is the noise parameter and $\|\bullet\|$ is the Euclidean distance between flows. This loss function supports the baseline detection that uses the Euclidean distance for anomaly judgement, which encourages instances with fewer shared microfeatures to be farther apart in the embedding space.

Since the interservice traffic has a large sequence of microelements, the loss function is constructed, and the error coefficients are computed to allow the model to learn larger interservice microelement features. The specific calculation equation is shown in Eq. (10)

$$\tau_2 = 0.5 \cdot \frac{|IP_S(S_x)\&IP_S(S_y)|}{24} + 0.5 \cdot \frac{|IP_D(S_x)\&IP_D(S_y)|}{24} \tag{10}$$

where mask denotes the sought IP binary code, IP_S denotes the source IP, and IP_D denotes the destination IP. This loss penalizes reliance on IP similarity, where traffic between identical or similar IP pairs is overfit to IP-specific patterns instead of generalizable microelement similarity.

If the number of clients for the same service is greater, the more accurate the micrometadata it creates. To reduce the noise of the model, the internode factor is calculated as shown in Eq. (11)

$$\tau_3 = 1/num(S_{ac}) + 1/num(S_{pe}) \tag{11}$$

where *num* is the number of client streams in the service. This loss function includes confidence-aware weighting based on microelement reliability, which reduces the penalty as the number of flows per service increases, encouraging the model to focus on microelements from abundant traffic samples.

In summary, the loss function is calculated as shown in Eq. (12)

$$\mathcal{L}(S_{an}, S_{ac}, S_{pe}) = \tau_2 * (\tau_1/\omega_1 + \omega_2 + \tau_3) \tag{12}$$

where ω_1 and ω_2 are regularization factors. ω_1 is set to the feature dimensionality to prevent loss explosion from high-dimensional vector norms. ω_2 is initialized as -1 to balance the scaling effect of τ_3 on the product of confidence factors. This ensures stable training and proper sensitivity to confidence variations across flows.

Our network architecture supports the representation of traffic relations, as motivated by the earlier discussion. For any two traffic flows, the learned embedding space satisfies the condition stated in Eq. 13

$$d(f_s - f_a) > d(f_d - f_a) \tag{13}$$

where d denotes the distance in the embedding space, f_a represents the anchor flow, f_s denotes the flow with a similar pattern to f_a, and f_d denotes the flow with a dissimilar pattern. The smaller distance between similar flows ensures that the model explicitly preserves the relational structure in the space, enabling effective discrimination between normal and anomalous traffic.

D. Traffic Anomaly Detection

We construct a traffic microelement spatiotemporal graph to represent the state. As illustrated in Fig. 12, IPs serve as nodes, and traffic flows as edges.

The graph is segmented by service. As the time window advances, we construct a baseline of normal traffic for each service within the time window. Specifically, we extract the

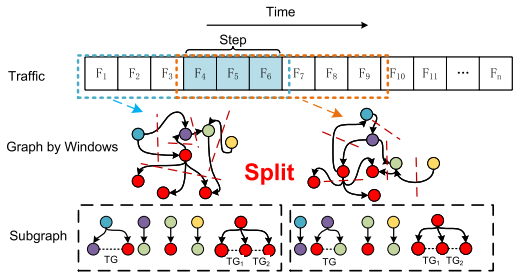


Fig. 12. Graphic of flow spatiotemporal behaviour.

spatiotemporal features of the traffic microelements in the subgraph for clustering according to Eq. (14)

$$F_n^i = \Psi_f(f_n^j\|) + \Psi_t(f) + \Psi_e(f) \tag{14}$$

where $\Psi_f(\bullet)$ represents the flow features, $\Psi_t(\bullet)$ represents the temporal feature and $\Psi_e(\bullet)$ represents the entropy feature. On this basis, clustering is performed using the K-means algorithm [55]. This process is formulated as shown in Eq. (15)

$$c^i = \underset{j}{argmin}(\left\|x^i - \mu_j\right\|), \mu_j = \frac{\sum_{i=1}^{m} 1\left\{c^i = j\right\} x^i}{\sum_{i=1}^{m} 1\left\{c^i = j\right\}} \tag{15}$$

where μ_j is the centre of the clusters and c^i represents the centre point speculation. The c^i of the majority clusters are utilized as the flow baselines for anomaly detection. Clustering is applied after the SNN, where the output consists of pairwise distance metrics between traffic flows. This naturally aligns with distance-based K-means methods, making them well suited for grouping flows by behavioural similarity.

In addition, we consider a particular server as an attack server where most of the traffic is attack traffic such that it is impossible to identify the flow baselines. Therefore, we also construct a baseline for the services and cluster for the service grouping. We extract features and perform K-means clustering for the server subgraph with the following features:

- Flow features. Central traffic features for flow baselines.
- Temporal features. Interactions of intragroup traffic within the service.
- Entropy features. Cross-entropy of client frequency.

Furthermore, we consider scenarios where services change frequently or attacks change the baselines. To address this, we employ a reputation-enhanced dynamic baseline construction method. First, the reputation scores are evenly allocated to all the baselines. When dynamic traffic is subsequently routed to a specific baseline, a corresponding reputation value is assigned to the client. If client traffic is segmented and allocated to a baseline, the client's reputation is updated and integrated into that baseline. Finally, all the reputation values are normalized to ensure consistency and fairness. The reputation calculation is shown as Eq. (16)

$$\begin{aligned} R_b &= \lambda(\alpha * R_n^{new} + (1-\alpha)R_n^{old}) \\ R_n &= \lambda(\beta * R_b + (1-\beta) * R_n^{old}) \end{aligned} \tag{16}$$

where R_b denotes the baseline reputation, R_n denotes the client reputation, α and β are the forgetting rates, and λ is the regularization function. The function λ is defined as shown in Eq. 17

$$\lambda(R_e) = mean(R_e) \tag{17}$$

where R_e denotes a reputation list that represents either baseline entries or the client.

E. Continuous Detection and Complexity Analysis

Continuous detection is achieved by maintaining a baseline repository for long-term anomaly monitoring. The process involves baseline construction and anomaly detection. When incoming traffic matches a known service in the database, its behaviour is compared against established baselines for anomaly identification. Concurrently, the reputation score is updated on the basis of the detection results, enabling adaptive refinement and improved baseline stability. For unseen services, new baselines are constructed and initialized on the fly. We conduct algorithm complexity analysis for real-time detection.

- Zeek processes N flows in $O(N)$. Subsequently, DBSCAN introduces $O(NlogN)$ overhead, where N denotes the total number of flows. Flow aggregation is incremental and lightweight.
- Greedy pairwise matching over K service-associated flows yields $O(KlogK)$ complexity.
- Sample selection and SNN inference both run in $O(K)$ because of parallel feature extraction and fixed architecture.
- Graph construction, K-means clustering, and dynamic baseline updates each take $O(K)$.

Overall, the pipeline achieves $O(N + NlogN + KlogK + K)$, dominated by $KlogK$, resulting in quasilinear complexity. The main limitation, which is greedy flow matching, is mitigated in practice, as most flows are sparse, reducing the number of effective comparisons. Consequently, the empirical runtime scales near $O(K)$, with sublinear growth under increasing load, confirming the high efficiency and scalability of real-time intrusion detection.

V. Experimental Analysis

A. Dataset and Experimental Setup

We use the CIC-IDS2017, CIC-IDS2018, and UNSW-NB15 datasets for evaluation. The nonencrypted flows in these datasets are processed using the same feature extraction pipeline as encrypted traffic, where plaintext content is not inspected, ensuring a protocol-agnostic evaluation framework.

The CIC-IDS2017 [50] dataset resembles real-world scenarios. To generate realistic traffic data, the dataset employs the B-Profile system to profile abstract human interaction behaviours and generate naturalistic benign background traffic. The dataset is based on 25 common user abstract behaviours built upon application layer protocols. It contains data from July 3 to July 7, 2017, spanning five days.

The CIC-IDS2018 [56] dataset is generated using a combination of B-profile and M-profile configuration files to create a diverse and comprehensive network intrusion detection dataset. Spanning a 10-day period in 2018, it captures a wide array of network activities. The attack infrastructure comprises 50 machines, and the victim organization has 5 departments and includes 420 machines and 30 servers.

The UNSW-NB15 [57] dataset is created by the IXIA PerfectStorm tool in the Cyber Range Lab of UNSW Canberra,

which generates a hybrid of real modern normal activities and synthetic contemporary attack behaviours.

The three public datasets contain pcap files of malicious traffic generated by network attacks, as well as features extracted for machine learning. Since the raw datasets are labelled only for features, we extract the traffic metadata from the pcap files and label them by comparison. Specifically, for the UNSW-NB15 dataset, the pcap files are labelled by 4-tuple. For the CIC-IDS2017 dataset, more precise labels are marked by matching the 4-tuple and timestamps of the flows. The CIC-IDS2018 dataset is labelled by combining the IPs of the attackers, the IPs of the victims, and the time period when the attack took place. The evaluation metrics are accuracy (AC), precision (PR), recall (RE), and F1 score (F1). Our code[3] is implemented using Zeek and Python.

To validate the effectiveness of the proposed algorithm, experiments are conducted on the same datasets and compared with seven state-of-the-art algorithms, representing three mainstream research directions. For the comparative algorithms, detection is performed using the corresponding feature types extracted from the datasets.

- Content-based detection. We choose the TMG-GAN algorithm as the baseline. The algorithm employs multiple generators to increase the diversity of generated samples, addressing the issue of sample imbalance in malicious traffic detection.
- Interaction-based detection. (1) DL-GNN [7] uses a dynamic line graph neural network to extract spatial information from each discrete snapshot and capture temporal variations in IP communication. It then uses line graphs to enhance the detection capabilities of the graph neural network. (2) ContraMTD [58] comprehensively considers both vertical and horizontal perspectives and uses SEC strategies and DE-GAT graph neural networks to extract local features and global interaction features of normal traffic. It employs contrastive learning techniques to learn the consistency relationship between these features and ultimately detects malicious traffic through multiple rounds of scoring.
- Anomaly-based detection. (1) CIC-GMM [50] uses a Gaussian mixture model for detection. (2) Rosett-IF [34] trains an isolation forest to classify the malicious data. (3) Kitsune [35] uses an autoencoder to train on normal data and identifies attack samples by evaluating the reconstruction error. (4) CPS-Guard [13] employs an outlier-aware autoencoder, detecting by setting an adaptive threshold and comparing reconstruction loss.

Parameter Settings. The training parameters for the SNN are configured as follows: 100 epochs, an initial learning rate of 0.001, and the Adam optimizer. The weights from the epoch with the lowest computed loss value are selected as the optimal model weights. The network architecture is configured as follows:

- The input layer is a linear layer with a size of $128 * 64$, the following is a batch normalization layer, and then a dropout layer with a dropout rate 0.2.
- The first hidden layer is a linear layer with a size of $64 * 32$, followed by a batch normalization layer and then a dropout layer with a dropout rate 0.2.
- The second hidden layer is a linear layer with a size of $32 * 16$.
- The output layer is a sigmoid layer, which limits the output values to the range $[0, 1]$. This facilitates the normalization of different types of classification outputs to a consistent data distribution range.

[3]https://github.com/gswsfh/BSTS-Net

B. Analysis of Experimental Results

Table IV presents the comparative experimental results between BSTS-Net and baseline algorithms across three datasets. Bold denotes the best results, and underline denotes the second-best results. The results indicate that BSTS-Net achieves the best detection performance across all three datasets, with F1 scores of 99.74, 99.63, and 99.88, respectively. Different sizes and categories of datasets demonstrate the effectiveness of BSTS-Net. Among the content-based methods, TMG-GAN achieves a high F1 on CICIDS2017 but F1 scores that are 3.07% and 6.85% lower on the others due to GAN training instability. Interaction-based DL-GNN and ContraMTD use GCNs and contrastive learning but struggle with irregular traffic. BSTS-Net avoids long-term dependencies by segmenting traffic into dynamic windows and comparing it against local baselines, achieving F1 values that are 8.84% and 11.92% higher than those of DL-GNN and ContraMTD on UNSW-NB15, respectively. Among the anomaly-based methods, Rosetta-IF achieves a 91% F1 on CICIDS2017 using TCP semantics, but BSTS-Net outperforms it by 17.21% via SNN-based microelement mapping. Kitsune and CPS-Guard use packet-level or environmental features, yielding only 54.99% and 59.70% F1 scores, respectively. CIC-GMM, which is based on Gaussian modelling, achieves an approximately 51% F1 and is sensitive to the data distribution.

C. Results of the per-Attack Binary Classification Experiment

In the binary classification experiment on the CICIDS2017 dataset, BSTS-Net achieves an F1 score exceeding 99% and outperforms the best baseline methods by 1.17% to 3.92% for the F1 and 3.13% to 3.99% for the accuracy across seven attack types. Simple anomaly-based baselines such as Kitsune have shown limited detection ability. The content-based methods perform better on Patator, Web Attack, Botnet, PortScan, and LOIT, achieving approximately 2% higher accuracy, but are less effective on DoS and DDoS attacks, with 1.85% lower accuracy and a 1.9% lower F1 score. Interaction-based methods struggle to capture structural patterns. BSTS-Net integrates traffic microelements and structural information, achieving the best overall performance and demonstrating strong robustness across diverse attack categories and scales.

D. Adversarial Experimental Analysis

To evaluate the detection capability under attack conditions, the dataset is perturbed, an adversarial dataset is constructed, and experiments are conducted accordingly. Specifically, the adversarial dataset is designed for the CICIDS2017 dataset, with packet-level, stream-level and feature-level perturbations constructed as follows:

- Packet perturbation. The packet-level perturbation is created by randomly adding, removing, or modifying a

TABLE IV
RESULTS OF THE MULTICLASSIFICATION EXPERIMENTS(%)

Datasets		CICIDS2017				CICIDS2018				UNSWNB15			
Model		AC	F1	NMI	AUC	AC	F1	NMI	AUC	AC	F1	NMI	AUC
Content-based	TMG-GAN	99.72	99.73	99.42	**99.98**	**99.87**	96.66	**99.57**	98.21	91.15	92.88	90.85	93.30
Interaction-based	DL-GNN	99.93	98.30	99.63	98.32	98.15	96.34	96.06	96.47	99.62	91.04	99.31	95.82
	ContraMTD	99.91	97.72	99.61	97.97	92.37	92.38	92.07	92.38	99.52	87.96	99.22	88.09
Anomaly-based	CIC-GMM	80.48	85.29	74.43	97.15	50.42	50.44	53.38	47.81	84.33	51.67	75.67	62.45
	Rosetta	95.25	91.34	87.12	91.56	85.74	83.49	80.42	83.64	95.21	81.73	79.48	81.30
	Kitsune	84.52	54.99	64.43	56.19	89.51	82.42	74.91	83.27	91.72	76.11	66.61	77.69
	CPS-Guard	87.86	60.67	61.54	60.78	61.47	62.57	63.98	62.61	84.54	59.70	61.55	59.75
Ours	BSTS-Net	**99.97**	**99.74**	**99.67**	99.80	99.65	**99.63**	99.55	**99.63**	**99.95**	**99.88**	**99.65**	**99.88**

TABLE V
BINARY CLASSIFICATION EXPERIMENTAL RESULTS(%)

Algorithms	CIC-GMM		Rosetta		Kitsune		CPS-Guard		TMG-GAN		DL-GNN		ContraMTD		BSTS-Net	
Attack Types	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1
Patator	50.23	50.48	83.54	83.51	82.48	82.49	62.50	62.41	96.65	96.69	96.65	96.59	90.02	90.27	**99.96**	**99.42**
Dos/DDos	51.63	51.41	80.20	80.19	87.95	88.03	65.09	65.11	94.81	94.75	96.66	96.65	92.55	92.63	**99.95**	**99.66**
Web Attack	48.48	48.52	79.61	79.62	86.41	86.45	59.55	59.25	96.06	96.06	95.79	95.73	92.15	92.39	**99.96**	**99.98**
Infiltration	48.51	48.56	83.47	83.50	80.27	79.85	61.58	61.33	96.15	96.16	94.87	97.79	91.92	91.94	**99.98**	**99.20**
Botnet	48.12	48.16	82.25	82.07	80.70	80.59	64.41	64.63	95.98	96.01	94.85	94.87	92.40	92.38	**99.97**	**98.42**
Port Scan	50.95	50.59	81.71	81.65	81.77	81.94	60.30	60.72	95.96	95.95	94.55	94.67	90.92	90.89	**98.86**	**97.12**
DDoS LOIT	50.67	50.77	85.37	85.29	91.91	82.09	63.42	63.06	96.62	96.65	94.65	94.72	91.67	91.79	**99.70**	**99.75**

TABLE VI
RESULTS OF THE ADVERSARIAL EXPERIMENTS (%)

	Anomaly-based								Content-based		Interaction-based				Ours	
Algorithms	CIC-GMM		Rosetta		Kitsune		CPS-Guard		TMG-GAN		DL-GNN		ContraMTD		BSTS-Net	
Disturbance type	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1
Packet perturbation	58.91	66.12	56.32	64.52	62.37	0.6983	65.56	53.68	83.02	82.47	85.74	86.48	88.87	88.67	97.85	96.84
Feature disguise	59.22	61.68	57.80	59.13	56.77	61.40	63.05	59.42	73.65	76.10	83.16	81.82	85.17	87.19	99.41	99.08
Flow disguise	82.34	84.29	89.76	89.66	69.85	58.74	75.03	67.04	87.96	87.85	77.75	80.34	78.45	79.24	98.04	98.07
None	80.48	85.29	95.25	91.34	84.52	54.99	87.76	60.67	99.72	99.73	99.93	98.30	99.91	97.72	99.97	99.74

certain number of packets within the flows of the original dataset.

- Flow perturbation. The flow-level perturbation is created by randomly adding, removing, or modifying a certain number of flows within the services of the original dataset.
- Feature perturbation. We modify the malicious flows such that their statistical features resemble those of the benign traffic in the dataset.

Table VI presents the adversarial evaluation results. BSTS-Net achieves the best performance across all the scenarios, with an F1 score decrease ranging from 0.66% to 2.9%. This strong robustness is due to its service-specific baseline construction from normal traffic, which reduces the impact of perturbations. In addition, the microelement space captures complex flow correlations, making it difficult for attackers to generate evasive samples. Interaction-based methods suffer heavily under flow perturbations, with F1 decreases of 17.96% and 18.48%, because injected noise disrupts the communication graph and impairs feature fusion. The content-based methods are most affected by feature perturbations, with a 23.63% decline in the F1 score as the adversarial samples resemble benign traffic. The anomaly-based baselines show inconsistent robustness, with F1 decreasing between 6.99% and 32.21% under packet and feature perturbations because of simplistic baseline modelling.

TABLE VII
BEHAVIOURAL MICROELEMENT INFORMATION

No.	Features	Description
1	4-tuple	SrcIP, SrcPort, DstIP, Dstport
2	microelements	5 types of traffic microelements
3	meta-information	Information of flows extracted by Zeek.

E. Traffic Microelement Analysis

Traffic microelements capture the interaction correlations between flows. To assess the consistency of the proposed microelements, we compare their features. The traffic information used for this analysis is detailed in Table VII. We analyse traffic consistency using metadata. First, flows sharing more than two identical microelements in either the source or the destination port are grouped. This grouping does not rely on IP addresses or service labels, making achieving accurate results more challenging. Next, we evaluate the consistency within groups by checking the metadata overlaps. The results across the three datasets are shown in Table VIII and demonstrate that more than 99 % of similar traffic from the same service is correctly aggregated, validating the ability of the microelements to capture flow relationships. This approach provides a novel and efficient feature extraction method. We also compare the consistency and efficiency of the microelements with those

TABLE VIII
FINGERPRINT CONSISTENCY DETERMINATION(%)

Datasets	IP Number	Consistency
CICIDS2017	19129	99.78%
CICIDS2018	92093	99.52%
UNSWNB15	22736	99.64%

TABLE IX
CICIDS2017 DATASET FINGERPRINT CONSISTENCY AND PERFORMANCE COMPARISON

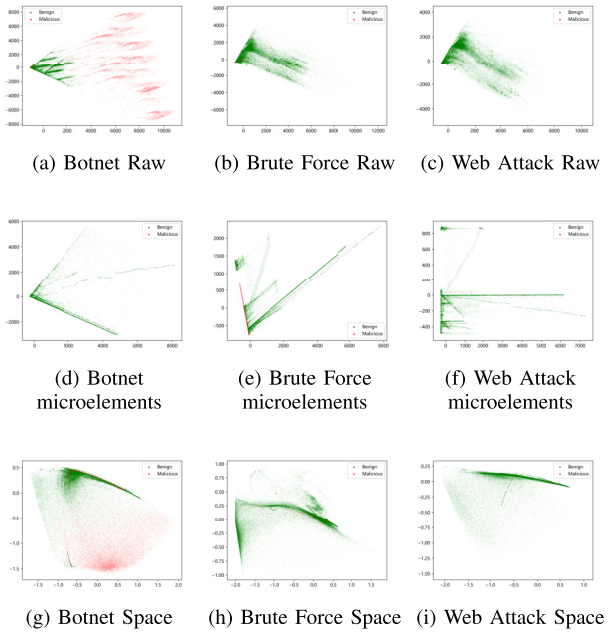
Method	Consistency	Time (s)
Microelements	99.78%	8
Statistical	74.28%	10
DL	28.64%	476
GNN	86.93%	384

of alternative methods, including statistical features, deep learning and graph neural networks. Deep learning consists of multiple fully connected layers. Graph neural networks employ two graph convolutional layers. Statistical features are extracted using CICFlowMeter [54]. The results are presented in Table IX. The microelement achieves the highest consistency and shortest processing, outperforming the worst method by 42.77% in terms of consistency and 47-fold in terms of efficiency. Deep learning performs poorly because of the randomness of the encrypted traffic. The use of a graph neural network improves the consistency by 58.29% when flow dependencies are modelled, but both deep and graph models require training and incur a computational cost that is more than an order of magnitude greater. Statistical features are efficient but sensitive to traffic variations, resulting in 25.5% lower consistency than microelements.

To study the distribution of the proposed traffic microelement space, we select samples from three major categories of attacks and label the samples simply as benign or malicious. Afterwards, we perform PCA dimensionality reduction on the original traffic metadata and traffic microelements for visualization. The results are shown in Fig. 13 (a-f). The experimental results indicate that although the original traffic shows some clustering indications of distribution, the distribution is irregular and inconsistent. In contrast, the traffic microelements exhibit a clear linear distribution, which demonstrates that the proposed microelements achieve a normalized suitable data distribution. However, relying solely on microelements overlooks the unique features of different samples, thereby causing diverse traffic patterns to be mapped into a nondiscriminative space. The traffic microelement space will be used to address this issue.

F. Analysis of the Traffic Microelement Space

To evaluate the mapping effectiveness of the microelement space, we visualize the same three attack features after mapping. Fig. 13 (g-i) reveals a surprising finding: despite the samples being input to the SNN consisting only of positive and negative sample pairs based on microelements, without explicit labels, benign and malicious traffic demonstrate distinct distribution patterns, especially for botnets. These patterns arise from the correlations and comparisons among the traffic microelements, thereby creating varying levels of correlation. Additionally, the samples are more uniformly



(a) Botnet Raw (b) Brute Force Raw (c) Web Attack Raw

(d) Botnet microelements (e) Brute Force microelements (f) Web Attack microelements

(g) Botnet Space (h) Brute Force Space (i) Web Attack Space

Fig. 13. Effect of feature extraction.

distributed, which minimizes both the false-positive rate for malicious traffic and the false-negative rate for benign traffic.

We conduct a comparative analysis of the training time and sample distribution, which reveals that the SNN reduces the loss to a low level within just 10 epochs, indicating fast convergence and minimal training overhead. Further experiments reveal that increasing the complexity of the SNN increases the training time without yielding significant improvements in detection performance, suggesting that a lightweight architecture is sufficient for effective learning.

G. Ablation Experiment

To investigate the contributions of each component of the BSTS-Net model, we conduct three ablation experiments.

- Ablation Experiment 1: Flow features are extracted by packet size and timestamps instead of traffic microelements.
- Ablation Experiment 2: The traffic microelement space is deleted, and the microelements are input into a spatiotemporal graph for classification.
- Ablation Experiment 2-1: The traffic microelement space is deleted and replaced with VAE embedding.
- Ablation Experiment 2-2: The traffic microelement space is deleted and replaced with GAE embedding.
- Ablation Experiment 3: The spatiotemporal graph is replaced with an autoencoder from the baseline methods.

The results of Table X on the CICIDS2017 dataset show that the traffic microelements significantly increase detection when relational patterns missed by traditional metrics are captured. The SNN enhances global pattern learning, but direct classification harms accuracy because of poor group generalization. Autoencoder-based methods underperform, with the GAE being better than the VAE. While autoencoder integration improves accuracy and recall, it remains sensitive to traffic

TABLE X
RESULTS(%) OF THE ABLATION EXPERIMENTS

Experiment	AC	F1	NMI	AUC
Ablation experiment 1	43.52	31.02	43.22	58.91
Ablation experiment 2	88.79	65.08	88.49	68.02
Ablation experiment 2-1	65.76	57.84	64.16	63.29
Ablation experiment 2-2	82.21	59.74	82.04	73.27
Ablation experiment 3	97.93	91.46	97.63	91.87
Original experiment	99.97	99.74	99.67	99.80

TABLE XI
RESULTS(%) OF THE FINE-GRAINED ABLATION EXPERIMENTS

Experiment	AC	F1	NMI	AUC
w/o $\vec{F}_f^r$	85.19	86.84	84.93	85.06
w/o $\vec{F}_f^l$	48.95	47.82	48.80	48.93
w/o $\vec{F}_f^p$	60.83	61.71	60.65	60.78
w/o $\vec{F}_f^c$	75.98	74.81	75.75	75.88
w/o $\vec{F}_f^d$	80.74	81.54	80.50	80.63
w/o τ_1	89.15	88.34	88.88	89.01
w/o τ_2	92.74	93.83	92.46	92.59
w/o τ_3	93.86	94.71	93.58	93.71
Original	99.97	99.74	99.67	99.80

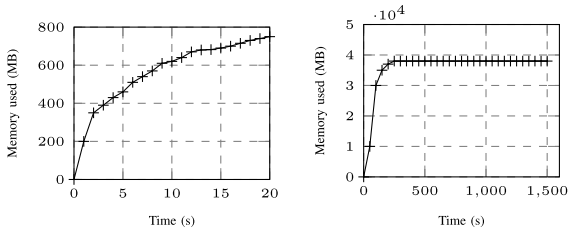
TABLE XII
RESULTS OF THE PERFORMANCE EXPERIMENTS(%)

Model	Training time(min)	Testing time(min)
TMG-GAN	184	2.8
DL-GNN	329	12
ContraMTD	897	65
CIC-GMM	7	0.14
Rosetta	2.7	0.09
Kitsune	74	0.6
CPS-Guard	56	0.4
BSTS-Net	18	0.3

noise, reducing precision. In addition, we conduct a fine-grained ablation study to evaluate the contribution of each of the five microelements and three components of the loss function. The results are presented in Table XI. Each feature and contrastive component is essential. $\vec{F}_f^l$ and $\vec{F}_f^p$ reduce the F1 scores by 51.92% and 38.03%, respectively, as they capture local patterns and directionality, which are core to protocol patterns. Among the loss components, τ_1 has the greatest impact, with an 18.2% decrease in the F1 score because of its focus on fine-grained low noise comparisons. In contrast, τ_2 and τ_3 yield smaller performance drops of 5.91% and 5.03%, respectively, as they operate over broader noisier traffic scopes with lower discriminability.

H. Performance Analysis

We evaluate the performance of the compared algorithms on the CICIDS2017 dataset. To ensure fairness, deep learning models are trained to convergence for optimal performance. The results are summarized in Table XII. The training and detection times of ContraMTD are the longest because of the use of dual-loop graph contrastive learning, taking 2.7 and 5.4 longer than those of the second-slowest method. DL-GNN scales superlinearly with the graph size, requiring 1.8 and



(a) Memory usage over time for BSTS. (b) Memory usage over time for AE-based method.

Fig. 14. Comparison of the memory usage over time.

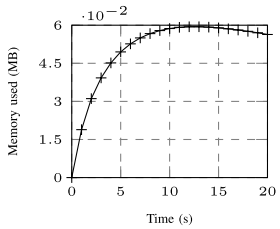


Fig. 15. Change in the proportion of baseline matches.

3.3 times more time than the third-slowest. method. TMG-GAN's adversarial training leads to a 2.5 higher training cost, although the inference is fast. Kitsune and CPS-Guard reduce inference latency but still require 3.1 and 1.3 times more time than the next slowest method, respectively. CIC-GMM and Rosett-IF complete detection in under 9 seconds but suffer up to a 46% decrease in the F1 score. In contrast, compared with the most expensive baseline, BSTS reduces the training time by more than 2.1-fold and achieves an inference that is 50% faster than the autoencoder-based models and only 12 seconds slower than the fastest method. As the baseline repository increases, the matching efficiency improves, enabling sublinear scaling and progressively lowering the end-to-end latency.

I. Online Analysis Experiments

BSTS-Net establishes service-specific baselines and monitors for deviations in real time. To evaluate its online detection capability, we replay CICIDS2017 traffic using tcpreplay. During processing, BSTS-Net dynamically generates 1317 baseline profiles. The corresponding memory and time overheads are shown in Fig. 14(a) and 14(b), respectively. BSTS-Net achieves real-time detection on a 2.7 GB dataset with a peak memory usage of 559 MB and an average of only 39 MB, with processing traffic at 145.6 MB/s. In contrast, the most memory-efficient autoencoder-based method requires a peak memory usage of 47189 MB, which is more than 84 times greater than that of BSTS-Net. This efficiency stems from the lightweight SNN and baseline-deviation classification. During detection, 65.84% of flows are classified via direct baseline matching, bypassing costly model inference. Fig. 15 shows that matching accelerates after an initial 20% slow phase caused by few baselines, revealing self-reinforcing efficiency.

J. Sensitivity Analysis

In this section, the CICIDS2017 dataset is used for experiments to validate the influence of parameter selection.

TABLE XIII
WINDOW SIZE ANALYSIS(%)

Window size	AC	F1	Cluster number	AC	F1
4	87.08	91.03	2	95.75	96.14
8	97.47	95.65	4	95.34	95.76
16	98.43	98.98	6	95.22	95.66
32	99.96	99.67	8	95.20	95.67
64	99.87	99.54	10	98.25	98.43
128	99.77	99.54	12	98.22	98.43
256	99.63	99.08	14	98.21	98.34

TABLE XIV
CONTRASTIVE LOSS HYPERPARAMETER ANALYSIS(%)

ω_1	AC	F1	ω_2	AC	F1
0.25*FeaDIM	95.86	89.52	-2	87.43	86.56
0.5*FeaDIM	93.54	94.78	-1.5	89.57	90.13
1*FeaDIM	99.97	99.74	-1	99.97	99.74
2*FeaDIM	74.82	75.18	-0.5	95.14	94.71
3*FeaDIM	59.86	60.41	0	97.74	96.85

1) Analysing the impact of window size selection. Given varying session lengths, the window size significantly affects classification. As shown in Table XIII, a window size of 4 is too small to capture traffic patterns, whereas a window size of 256 includes mixed communication stages and introduces interference. For the dataset in this work, a window size of 32 achieves optimal performance.
2) Analysing the impact of cluster size selection. As shown in Table XIII, small cluster sizes disperse malicious samples across clusters, increasing the false-positive rate. A cluster size of 10 achieves optimal detection performance, with stability maintained for larger sizes and no significant degradation observed.

In conclusion, the selection of parameters in this study influences the experimental outcomes. However, window sizes ranging from 8 to 256 and cluster numbers from 6 to 14 yield relatively high accuracy. The robustness of the proposed model is insensitive to parameter variations.

In addition, we analyse the impact of hyperparameter settings in contrastive loss on detection performance. The results are summarized in Table XIV. A large ω_1 weakens the relation modelling and sharply decreases the F1 score. A small ω_1 causes overfitting to host-specific patterns. A large ω_2 amplifies noise and false positives. A small ω_2 fails to balance the confidence weighting. ω_1 has a greater impact than ω_2 because it nonlinearly scales an unstable loss term. Training is highly sensitive to ω_1, and improper values cause severe degradation.

K. Zero-Day Attack Detection

BSTS-Net effectively detects zero-day attacks by modelling robust benign baselines and identifying deviations. When trained on CICIDS2017 and evaluated on CICIDS2018, it achieves 99.59% recall, demonstrating strong generalizability to unseen attacks. However, when applied to the benign traffic of CICIDS2018, a 13.84% false-positive rate occurs because of behavioural shifts not captured by the static baseline. This highlights the need for adaptive baseline modelling in new environments. While our current approach excels in stable settings, we plan to develop dynamic updating mechanisms to improve generalization and reduce false alarms in evolving networks.

VI. CONCLUSION

Current malware detection methods rely on payload or traffic interactions, which are easily evaded by encryption. In this paper, BSTS-Net, a behaviour-based approach robust to encrypted traffic, is proposed. We introduce a greedy-based microelement extraction method to capture intrinsic traffic relations. An SNN models local and global communication patterns in the microelement space without the need for labels. We further construct a spatiotemporal graph integrated with reputation scoring for dynamic anomaly detection. Clustering and dynamic baseline analysis enable effective malicious traffic identification. Experiments show that BSTS-Net achieves detection rates over 99% across multiple datasets, outperforming the state-of-the-art methods by up to 8%, with strong robustness under adversarial settings. Future work will explore richer microelement representations.

REFERENCES

[1] S. Elmohamed. (2021). *Acronis Cyberthreats Report 2021*. [Online]. Available: https://community.spiceworks.com/t/acronis-cyberthreats-report-mid-year-2021/807520
[2] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan, "HOLMES: Real-time APT detection through correlation of suspicious information flows," in *Proc. IEEE Symp. Security Privacy*, Mar. 2019, pp. 1137–1152.
[3] (2024). *Suricata 7.0.6*. [Online]. Available: https://suricata.io/
[4] (2024). *Snort 3*. [Online]. Available: https://www.snort.org/
[5] R. A. Bridges, T. R. Glass-Vanderlan, M. D. Iannacone, M. S. Vincent, and Q. Chen, "A survey of intrusion detection systems leveraging host data," *ACM Comput. Surv. (CSUR)*, vol. 52, no. 6, pp. 1–35, 2019.
[6] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5998–6008.
[7] H. Ding, Y. Sun, N. Huang, Z. Shen, and X. Cui, "TMG-GAN: Generative adversarial networks-based imbalanced learning for network intrusion detection," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 1156–1167, 2024.
[8] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2008.
[9] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 2, pp. 494–514, Feb. 2022.
[10] Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang, "A survey on neural network interpretability," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 5, no. 5, pp. 726–742, Oct. 2021.
[11] M. Liu et al., "Enhanced detection of obfuscated HTTPS tunnel traffic using heterogeneous information network," *Comput. Netw.*, vol. 257, Sep. 2024, Art. no. 110975.
[12] V. Jyothsna, V. V. R. Prasad, and K. M. Prasad, "A review of anomaly based intrusion detection systems," *Int. J. Comput. Appl.*, vol. 28, no. 7, pp. 26–35, 2011.
[13] M. Catillo, A. Pecchia, and U. Villano, "CPS-GUARD: Intrusion detection for cyber-physical systems and IoT devices using outlier-aware deep autoencoders," *Comput. Secur.*, vol. 129, Jun. 2023, Art. no. 103210.
[14] D. Zhao et al., "Botnet detection based on traffic behavior analysis and flow intervals," *Comput. Secur.*, vol. 39, pp. 2–16, Nov. 2013.
[15] X. Chen and K. He, "Exploring simple Siamese representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 15750–15758.
[16] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Da Nang, Vietnam, Jan. 2017, pp. 712–717.

[17] R.-H. Hwang, M.-C. Peng, V.-L. Nguyen, and Y.-L. Chang, "An LSTM-based deep learning approach for classifying malicious traffic at the packet level," *Appl. Sci.*, vol. 9, no. 16, p. 3414, Aug. 2019.
[18] X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi, and J. Yu, "ET-BERT: A contextualized datagram representation with pre-training transformers for encrypted traffic classification," in *Proc. ACM Web Conf.*, Apr. 2022, pp. 633–642.
[19] G. Zhou, X. Guo, Z. Liu, T. Li, Q. Li, and K. Xu, "TrafficFormer: An efficient pre-trained model for traffic data," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2025, pp. 1844–1860.
[20] C. Fu, Q. Li, M. Shen, and K. Xu, "Realtime robust malicious traffic detection via frequency domain analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2021, pp. 3431–3446.
[21] D. Barradas, N. Santos, L. Rodrigues, S. Signorello, F. M. V. Ramos, and A. Madeira, "FlowLens: Enabling efficient flow classification for ML-based network security applications," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, Mar. 2021, pp. 4–5.
[22] J. Holland, P. Schmitt, N. Feamster, and P. Mittal, "New directions in automated traffic analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2021, pp. 3366–3383.
[23] G. Zhou, Z. Liu, C. Fu, Q. Li, and K. Xu, "An efficient design of intelligent network data plane," in *Proc. 32nd USENIX Secur. Symp.*, 2023, pp. 6203–6220.
[24] Z. Zhao, Z. Li, Z. Song, W. Li, and F. Zhang, "Trident: A universal framework for fine-grained and class-incremental unknown traffic detection," in *Proc. ACM Web Conf.*, May 2024, pp. 1608–1619.
[25] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting botnet command and control channels in network traffic," in *Proc. 15th Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2008, pp. 4–5. [Online]. Available: https://api.semanticscholar.org/CorpusID:13302792
[26] M. Shen, Y. Liu, L. Zhu, X. Du, and J. Hu, "Fine-grained webpage fingerprinting using only packet length information of encrypted traffic," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 2046–2059, 2021.
[27] C. Fu, Q. Li, M. Shen, and K. Xu, "Detecting tunneled flooding traffic via deep semantic analysis of packet length patterns," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Dec. 2024, pp. 3659–3673.
[28] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.
[29] A. Pareja et al., "EvolveGCN: Evolving graph convolutional networks for dynamic graphs," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 4, pp. 5363–5370.
[30] G. Duan, H. Lv, H. Wang, and G. Feng, "Application of a dynamic line graph neural network for intrusion detection with semisupervised learning," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 699–714, 2023.
[31] X. Li et al., "A high accuracy and adaptive anomaly detection model with dual-domain graph convolutional network for insider threat detection," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1638–1652, 2023.
[32] Y. Zhong, Z. Wang, X. Shi, J. Yang, and K. Li, "RFG-HELAD: A robust fine-grained network traffic anomaly detection model based on heterogeneous ensemble learning," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 5895–5910, 2024.
[33] C. Fu, Q. Li, and K. Xu, "Detecting unknown encrypted malicious traffic in real time via flow interaction graph analysis," 2023, *arXiv:2301.13686*.
[34] R. Xie et al., "Rosetta: Enabling robust TLS encrypted traffic classification in diverse network environments with TCP-aware traffic augmentation," in *Proc. ACM Turing Award Celebration Conf.-China*, Jul. 2023, pp. 131–132.
[35] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," 2018, *arXiv:1802.09089*.
[36] M. Nakıp and E. Gelenbe, "Online self-supervised deep learning for intrusion detection systems," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 5668–5683, 2024.
[37] E. Papadogiannaki and S. Ioannidis, "A survey on encrypted network traffic analysis applications, techniques, and countermeasures," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 1–35, Jul. 2022.
[38] K. J. Sullivan, W. G. Griswold, Y. Cai, and B. Hallen, "The structure and value of modularity in software design," *ACM SIGSOFT Softw. Eng. Notes*, vol. 26, no. 5, pp. 99–108, Sep. 2001.
[39] M. Papazoglou and W. Van Den Heuvel, "Service oriented architectures: Approaches, technologies and research issues," *VLDB J.*, vol. 16, no. 3, pp. 389–415, 2007.
[40] I. Nadareishvili, R. Mitra, M. McLarty, and M. Amundsen, *Microservice Architecture: Aligning Principles, Practices, and Culture*. Sebastopol, CA, USA: O'Reilly Media, 2016.
[41] O. Mämmelä, J. Hiltunen, J. Suomalainen, K. Ahola, P. Mannersalo, and J. Vehkaperä, "Towards micro-segmentation in 5G network security," in *Proc. Eur. Conf. Netw. Commun. (EuCNC) Workshop Netw. Manage.*, 2016, pp. 3–5.
[42] S. McCanne et al., "Toward a common infrastructure for multimedia-networking middleware," in *Proc. 7th Int. Workshop Netw. Operating Syst. Support Digit. Audio Video (NOSSDAV)*, Jun. 1997, pp. 39–49.
[43] Y. Fu, H. Xiong, X. Lu, J. Yang, and C. Chen, "Service usage classification with encrypted internet traffic in mobile messaging apps," *IEEE Trans. Mobile Comput.*, vol. 15, no. 11, pp. 2851–2864, Nov. 2016.
[44] M. Korczyński and A. Duda, "Markov chain fingerprinting to classify encrypted traffic," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2014, pp. 781–789.
[45] (2024). *Zeek 7.0*. [Online]. Available: https://zeek.org/about/
[46] B.-K. Lee and J.-H. Chang, "Packet loss concealment based on deep neural networks for digital speech transmission," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 24, no. 2, pp. 378–387, Feb. 2016.
[47] (2024). *Service Name and Transport Protocol Port Number Registry*. [Online]. Available: http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml
[48] G. Peng, "CDN: Content distribution network," 2004, *arXiv:cs/0411069*.
[49] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN," *ACM Trans. Database Syst.*, vol. 42, no. 3, pp. 1–21, Sep. 2017.
[50] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, Portugal, Jan. 2018, pp. 108–116.
[51] (2024). *Samba 4.20.2 Document*. [Online]. Available: https://www.samba.org/samba/docs/
[52] (2013). *Multicast DNS*. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6762
[53] (2000). *Http Over TlS*. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2818
[54] (2021). *Ahlashkari/Cicflowmeter*. [Online]. Available: https://github.com/ahlashkari/CICFlowMeter
[55] M. Ahmed, R. Seraj, and S. M. S. Islam, "The k-means algorithm: A comprehensive survey and performance evaluation," *Electronics*, vol. 9, no. 8, p. 1295, Aug. 2020.
[56] (2018). *A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018)*. [Online]. Available: https://registry.opendata.aws/cse-cic-ids2018/
[57] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.
[58] X. Han et al., "ContraMTD: An unsupervised malicious network traffic detection method based on contrastive learning," in *Proc. ACM Web Conf.*, May 2024, pp. 1680–1689.