RESEARCH-ARTICLE

FlowSentry: Accelerating NetFlow-based DDoS Detection

XIAOYU HE, Tsinghua University, Beijing, China

XIAOHUI XIE, Tsinghua University, Beijing, China

XIN WANG, Stony Brook University, Stony Brook, NY, United States

LEI ZHANG, Beijing Institute of Technology, Beijing, China

KUN XIE, Hunan University, Changsha, Hunan, China

LIN CHEN, China Telecom Corporation Limited, Beijing, Beijing, China

View all

Published: 19 November 2025

Citation in BibTeX format

CCS '25: ACM SIGSAC Conference on Computer and Communications Security
October 13 - 17, 2025
Taipei, Taiwan

Conference Sponsors:
SIGSAC

https://doi.org/10.1145/3719027.3744800
ISBN: 9798400715259

FlowSentry: Accelerating NetFlow-based DDoS Detection

Xiaoyu He
Tsinghua University
Beijing, China

Xiaohui Xie*
Tsinghua University
Beijing, China

Xin Wang
Stony Brook University
Stony Brook, United States

Lei Zhang*
Zhongguancun Laboratory
Beijing, China

Kun Xie
Hunan University
Changsha, China

Lin Chen
China Telecom Cybersecurity Technology Co.,Ltd.
Beijing, China

Yong Cui*
Tsinghua University
Beijing, China

Abstract

Distributed Denial of Service (DDoS) attacks threaten the stability of online services by overwhelming them with excessive traffic. NetFlow-based DDoS detection systems are widely adopted by Internet Service Providers (ISPs) in upstream multi-point detection scenarios to provide robust detection for volumetric DDoS attacks. However, these systems face inherent delays, as NetFlow detection is non-instantaneous—routers aggregate and summarize flow records over a period before reporting, which impacts timely detection. Existing research primarily focuses on optimizing the NetFlow reporting mechanism at the router side. Unfortunately, the need for either software or hardware upgrades for routers would incur a high deployment cost, which is impractical for ISPs in the short term. In this paper, we propose FlowSentry, a novel NetFlow detection framework to accelerate DDoS attack identification at the server side. The system operates on a dual-layer filtering paradigm to handle the high-frequency NetFlow records, incorporating two core technologies: ADWindow and STAnalyzer. ADWindow is a sketch-based sliding window mechanism designed to retain possibly anomalous flow information, filtering out benign flows to reduce the computational overhead. STAnalyzer leverages the cross-router traffic correlation to efficiently infer abnormal growth patterns of potential malicious traffic based on partially reported flow records, thus significantly reducing the detection delay. Our extensive experiments in simulated backbone network environments demonstrate that FlowSentry achieves better detection accuracy while reducing the detection delay by up to 65.63% compared to existing methods.

CCS Concepts

• **Networks** → **Denial-of-service attacks**; *Network monitoring*; • **Security and privacy** → **Denial-of-service attacks**.

*Yong Cui, Xiaohui Xie and Lei Zhang are the corresponding authors.

CCS '25, Taipei.

ACM ISBN 979-8-4007-1525-9/2025/10
https://doi.org/10.1145/3719027.3744800

Keywords

DDoS Detection, NetFlow, Sketch, Backbone Network

ACM Reference Format:
Xiaoyu He, Xiaohui Xie, Xin Wang, Lei Zhang, Kun Xie, Lin Chen, and Yong Cui. 2025. FlowSentry: Accelerating NetFlow-based DDoS Detection. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), October 13–17, 2025, Taipei.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3719027.3744800

1 Introduction

Distributed Denial of Service (DDoS) attacks represent a formidable threat to the stability and availability of online services worldwide [4, 9, 41]. In recent years, volumetric DDoS attacks have grown in scale, frequency and complexity [2, 15, 16]. In these attacks, perpetrators leverage distributed networks of compromised systems to flood victims' upstream links with traffic up to 5.6 Tbps, rendering services unavailable to legitimate users [9]. Recent industry reports show that volumetric attacks are escalating at an unprecedented pace, capable of reaching 500 Gbps within just 2 seconds and scaling up to 1 Tbps in 10 seconds [10, 20]. Furthermore, these attacks are becoming increasingly short-lived, with over 30% lasting less than 5 minutes and 55% under 10 minutes. To effectively counteract these high-velocity threats, there is a critical need for detection systems to be robust, rapid, and precise.

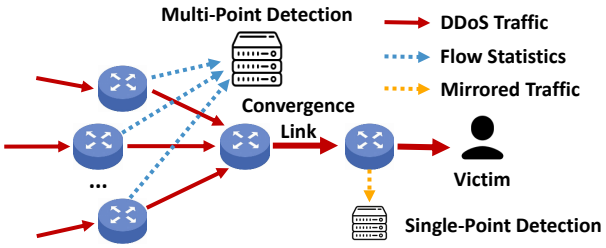


Figure 1: Single-Point vs. Multi-Point DDoS Detection.

As shown in Fig. 1, current DDoS detection can be categorized into single-point and multi-point approaches based on the detection location relative to the DDoS traffic convergence link. Single-point detection operates downstream of the convergence link, close to the entry point of the victim's network, where raw traffic is mirrored and analyzed at the packet level [5, 12–14, 29, 35]. However, such detection is inherently fragile: when an attack rapidly congests the convergence link, the downstream detector can only receive a portion of the traffic, resulting in performance degradation. To achieve

robust post-attack detection, it is critical to deploy multi-point detection upstream of the convergence link, which is generally carried out by Internet Service Providers (ISPs) through the deployment of NetFlow-like monitoring systems (e.g., NetStream [19], J-Flow [21]). These systems involve routers, typically boundary routers of backbone networks to better cover a larger volume of traffic [34], to sample critical flow information such as packet counts and byte counts. The sampled flow records are periodically forwarded to centralized detection servers using a timeout-based reporting mechanism [8].

To manage the large-scale traffic in backbone networks, NetFlow routers adopt relatively long timeout reporting periods, typically ranging from 1 to 5 minutes. Such reporting delays are a major limitation to detection efficiency, particularly in responding to fast-moving volumetric DDoS attacks [35]. Current efforts primarily focus on router-side optimizations, including generating flexible flow records and offloading parts of detection tasks to routers [7, 33, 36, 38, 43]. However, these approaches require either hardware or software upgrades to routers, such as the deployment of high-performance programmable devices and the implementation of new protocols. For ISPs that prioritize low-cost and stable operations, such upgrades are largely impractical. The high costs of programmable switches required for PB-scale traffic present a significant financial barrier. Furthermore, the complexity and risks of debugging and coordinating upgrades across hundreds of core routers make such efforts even less feasible.

Thus, we provide a different and complementary solution: rather than making costly and complex router-side upgrades, we dedicate efforts to leveraging NetFlow server-side optimizations to improve detection timeliness. Our key insights lie in two critical observations in practice. First, the reporting times of NetFlow routers are not synchronized due to the timeout reporting mechanism [8]. Accordingly, from the detection server's perspective, sampled attack flow records accumulate incrementally as a DDoS attack unfolds. This provides an opportunity for early anomaly detection by leveraging partially reported flow information. Second, volumetric DDoS attacks are typically coordinated across multiple distributed sources, resulting in strong spatiotemporal dependencies as malicious traffic converges through several boundary routers. This coordinated behavior creates noticeable correlation patterns across boundary routers. Additionally, the reuse of botnets and attack scripts further amplifies this regularity, as similar traffic patterns are reproduced across different attack campaigns and locations. In a nutshell, we aim to learn the cross-router correlations to forecast the overall traffic growth using only a subset of reported flow records.

However, establishing a prediction-based detection mechanism comes with significant challenges. First, the massive volume of flow records in backbone networks presents a computational bottleneck to the detection system. NetFlow routers can report up to 1.3 million flows per second. Existing server infrastructures often lack the capacity to execute high-frequency prediction models at this scale. Second, accurately estimating traffic growth based on network flow data is inherently difficult. Traffic prediction is a well-known, long-standing problem, even with fine-grained packet-level data. When relying on coarse-grained and sampled NetFlow records, the challenge is magnified, as critical details about traffic dynamics are lost during the sampling process.

We address these challenges and propose FlowSentry, a multi-point DDoS detection framework that accelerates detection and operates on existing NetFlow servers without requiring router modifications. Operating on a dual-layer filtering paradigm, FlowSentry aims to condense extraneous data to reduce the computational overhead while amplifying the detection sensitivity towards potentially malicious flows. The system integrates two core technologies: ADWindow (short for Anomaly Detection Window) and STAnalyzer (short for Spatiotemporal Analyzer). As a sliding window mechanism, ADWindow is designed to process incoming flow records at high frequency. It adopts sketch technology to efficiently sift and retain information pertaining to potentially anomalous flows within a defined time window. Leveraging an anomaly scoring function, ADWindow probabilistically replaces older flow attributes with newer ones based on their assessed levels of suspiciousness. This approach efficiently filters out a large volume of normal traffic. STAnalyzer circumvents the need for precise predictions by focusing on identifying whether traffic growth exceeds an anomaly threshold. It leverages the cross-router traffic correlation to roughly infer overall network dynamics. In the offline phase, it trains a Vector Autoregression (VAR) model on labeled historical flow data to learn the growth patterns of large flows traverse multiple routers, including both normal traffic and DDoS traffic. Router communities are formed using a community detection algorithm, enabling the model to focus on meaningful interdependencies across routers. In the online phase, STAnalyzer processes flow records pre-screened by ADWindow. Using the pre-trained VAR model, it analyzes real-time data to infer traffic trends and detect anomalies.

In summary, this work makes 3 main contributions:

(1) We propose FlowSentry, a NetFlow server-side detection framework designed to enable the rapid identification of DDoS attacks. FlowSentry exploits the asynchronous reporting of NetFlow routers and the spatiotemporal regularity introduced by coordinated botnet-based attacks. It employs a dual-layer filtering paradigm to streamline data processing by eliminating extraneous traffic.
(2) We design ADWindow to perform efficient initial screening within FlowSentry. ADWindow compactly retains information on potentially anomalous flows, addressing the challenge of computational overhead posed by the high volume of flow records.
(3) We develop STAnalyzer to perform thorough re-screening within FlowSentry. STAnalyzer groups routers into communities, uses a Vector Autoregression (VAR) model to learn growth patterns of cross-router large flows, and predicts overall traffic trends based on partially reported flow data to accurately identify anomalies.

The performance of FlowSentry is evaluated in a simulated NetFlow detection system using one public dataset and two real-world datasets. The experimental results demonstrate that FlowSentry achieves a significant reduction in detection delay, averaging a 65.63% decrease compared to the top-performing baseline. Additionally, FlowSentry enhances the detection accuracy, achieving an overall increase in AUC by 0.65%, an F1 score improvement of 0.92%, and a 6.44% reduction in FPR.

2 Background and Motivation

2.1 NetFlow-based Detection System

NetFlow is a widely deployed Cisco IOS technology for the collection and export of IP flow information [8, 26]. As the standard for obtaining operational data from IP networks, it supports critical functions such as network monitoring and traffic analysis. The exported flow data, known as a NetFlow record, contains key statistics detailed in Table 1. To manage the massive volume of traffic in large-scale backbone networks, NetFlow employs a sampling strategy, often ranging from 5000:1 to 1000:1 in sequential sampling.

Table 1: NetFlow Record Main Information.

Feature	Description	Size
srcIP	Source IP address of the flow	32/128 bits
dstIP	Destination IP address of the flow	32/128 bits
sport	Source port of the flow	16 bits
dport	Destination port of the flow	16 bits
protocol	IP protocol value	8 bits
packet count	Total number of packets in the flow	32 bits
byte count	Flow length in bytes	32 bits
start time	Start time of the flow	64 bits
end time	End time of the flow	64 bits
routerIP	IP address of the router that reported the flow	32/128 bits

As shown in Fig. 2(a), a NetFlow-based detection system operates through multiple boundary routers (typically ranging from 60 to 200 in a backbone network) that collect the flow information in NetFlow caches and periodically transmit the data to a centralized detection server through a timeout reporting mechanism. Timeout reporting includes two types: active timeout and inactive timeout.
- **Active Timeout**: Routers periodically flush the cached flow records and report them at regular intervals, typically every 60 to 300 seconds depending on the traffic conditions. This interval is chosen to trade off between reducing the reporting overhead and ensuring timely updates for network monitoring.
- **Inactive Timeout**: An inactive timeout can be triggered under circumstances such as when the NetFlow cache is full or when a specific flow remains inactive for a predefined duration. For further details, please refer to [8].

Asynchronous Flow Reporting. Due to factors like message sending upon reaching the cache capacity and inherent monitoring policies, the reporting times of boundary routers are often asynchronous and difficult to fully synchronize. This lack of synchronization means that flow records from different routers are reported at varying intervals. Therefore, after an attack occurs, the detector receives attack flow records gradually. As illustrated in Fig. 2(b), each box represents the flow information recorded over one reporting cycle, which is only transmitted to the detector at the end of that cycle. In this example, Router 1 and Router 2 send their reports earlier, but the detector can only identify the attack scale after receiving the first report from Router 3. Furthermore, it is not until Router 3's subsequent report that the detector obtains a complete view of the attack's full scope and scale. This staggered reporting mechanism, intrinsic to the NetFlow protocol, introduces a delay in accurately assessing and responding to network attacks [17, 35].

2.2 Motivation

In this section, we provide the key motivations for this work from three perspectives.

1. Why Multi-Point Detection is Essential:

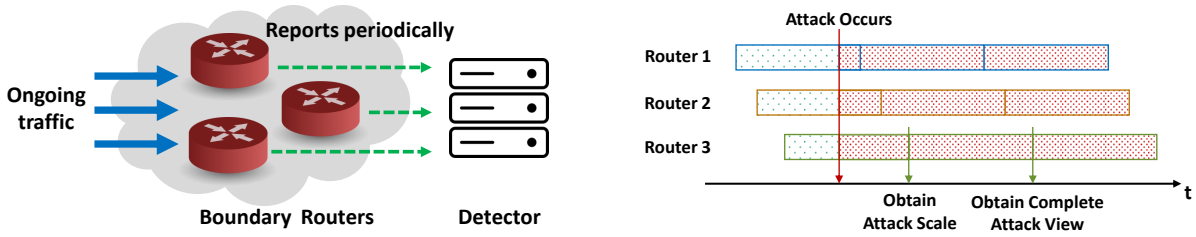
Volumetric DDoS attacks are characterized by ultra-large scale activities, with traffic surging within seconds and ramp-up speeds reaching record highs. Attackers often exploit amplifiers across the entire network to synchronize and generate massive traffic volumes, rapidly occupying the upstream bottleneck links of the victims with sudden spikes [6, 15].

The performance of downstream single-point detection systems suffers in the face of such volumetric attacks. The severe packet loss and latency resulting from the throttling of the network bottleneck disrupt the timely delivery of flow records to detectors, significantly delaying or degrading their ability to respond effectively to DDoS attacks. Moreover, only monitoring traffic near the target network's entry point, single-point detection systems have difficulty in handling distributed and large-scale attacks. In contrast, upstream multi-point detection systems offer a more robust solution when facing rapidly escalating volumetric DDoS attacks. A prominent example is backbone network detection. Backbone networks, with their high bandwidth capacity, redundant architecture, and advanced traffic management systems, are naturally resistant to volumetric attacks. Positioned centrally, backbone networks can well monitor the global traffic patterns and provide a comprehensive overview that single-point systems cannot achieve. Furthermore, the widespread deployment of NetFlow detection systems across the boundary routers of backbone networks enhances traffic flow coordination and analysis.

2. Why NetFlow-based Detection Needs Improvement:

NetFlow-based detection involves routers collecting flow information over a preset interval (i.e., NetFlow active timeout) and then reporting aggregated flow statistics to a central detection system for analysis. To manage the large-scale traffic typical of backbone networks, this interval is set to be relatively long, ranging from one to five minutes. Consequently, when an attack occurs, the detector cannot receive the attack flow records in real time. The flow records are gradually received over a collection period, as discussed in Section 2.1, resulting in inherent detection delays.

Current industrial practice fails to address the delay issue. It employs a batch-based detection framework that waits for all boundary routers to report flow records before conducting a unified detection process, typically performed once per reporting cycle. This approach is driven by practical considerations: ensuring network-wide attack visibility while keeping computational costs manageable for ISPs. However, the need to wait for complete flow reports before analysis prolongs the detection cycle, making such batch-based detection unsuited for rapid, short-duration volumetric attacks.



(a) NetFlow Reporting Process (b) Sequential Attack Detection Timeline

Figure 2: NetFlow timeout reporting mechanism.

3. Why Router-side Methods Are Impractical:

Upstream multi-point detection systems are typically deployed and managed by ISPs to monitor and mitigate DDoS attacks across their backbone networks. For ISPs, the primary goal is to minimize costs while ensuring stable and reliable service. However, implementing router modifications [33, 36] is often impractical due to cost and deployment difficulties [32].

Compared to fixed-function router solutions, a much higher cost will be incurred to replace existing hardware with advanced, programmable network devices to enhance flow statistics at the router level. In addition to the hardware costs, developing and integrating custom software to fully utilize these devices' capabilities can be both time-consuming and resource-intensive. Specialized expertise is often required, further escalating operational expenses.

Moreover, the deployment of such systems can disrupt network stability, leading to significant operational challenges. New hardware and software installations often result in network downtime, which can degrade service quality and erode customer trust. The added complexity of training network administrators and engineers to manage the new infrastructure also contributes to the potential for operational disruptions. These issues undermine the stability that ISPs strive for and add risks to maintaining service reliability.

2.3 Problem Scope

We focus on deploying the DDoS detector at the NetFlow server, without requiring any modifications to the router-side sampling and reporting mechanisms. The approach involves offline training with historical labeled data, followed by real-time detection on live NetFlow record streams. The detector could further integrate with mitigation devices or IP blocklists to automate defense responses.

Our research specifically targets volumetric DDoS attacks that consume a significant amount of bandwidth and are typically detectable through NetFlow data. Attackers are capable of launching various types of DDoS attacks that last for only a limited duration. Our study does not address advanced low-bandwidth attack strategies, including low-rate DDoS [24], application-layer attacks such as Slowloris, or link-flooding attacks [22], as such low-volume attack flows are often not captured by NetFlow's sampling mechanisms.

3 Overview of FlowSentry

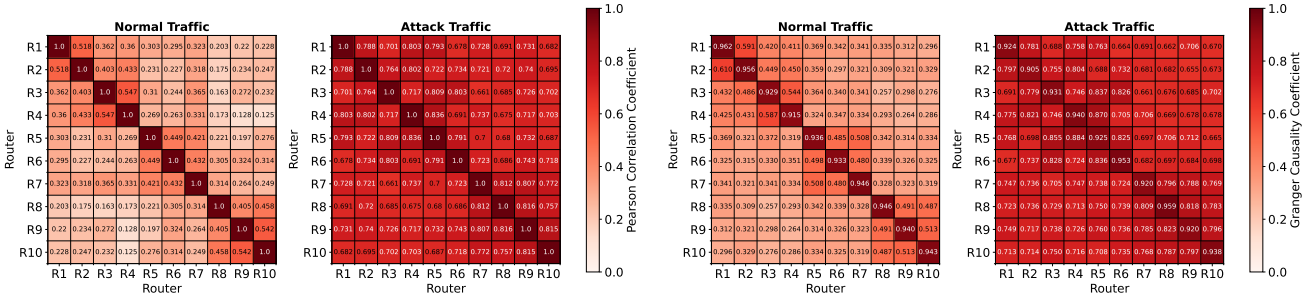
We present FlowSentry, a NetFlow-based detection framework designed for rapid and accurate identification of DDoS attacks at the server end. As previously discussed, a key enabler of FlowSentry's efficiency is its ability to predict potential DDoS attacks using a minimal number of flow records, thereby significantly accelerating the detection process.

Accurately estimating traffic growth has long been a challenging problem, particularly when relying on limited, coarse-grained NetFlow data. However, our goal does not require precise predictions for all flows. Instead, we focus on determining whether the traffic is likely to exhibit anomalous growth. By shifting from exact estimation to anomaly detection, we substantially reduce the complexity of the problem while retaining the predictive power necessary for effective DDoS identification.

To achieve this goal, we leverage the spatiotemporal correlations present in inter-router network traffic. These correlations emerge from two key factors: network management mechanisms and DDoS attack behaviors. For network management, routing policies and traffic engineering mechanisms [3] influence how traffic flows through the network, shaping the spatiotemporal relationships between routers. The correlations are particularly evident in strategies like load balancing, where traffic is dynamically distributed across multiple routers situated between metropolitan and backbone networks. This distribution not only optimizes resource utilization and mitigates congestion but also gives rise to predictable traffic patterns that reflect real-time network demand and fluctuations. For attack behaviors, DDoS attacks exhibit strong spatiotemporal dependencies. Primarily, the coordination of the attack, where malicious traffic is synchronized across multiple sources, creates spatiotemporal dependencies as the traffic flows through several boundary routers. Secondarily, the reuse of botnets and attack scripts leads to predictable patterns, as the same attack methods are repeatedly employed across different attack campaigns and geographic regions. Our approach is built on the hypothesis that these correlations can serve as early indicators of DDoS activity. By focusing on these correlations, we can efficiently detect anomalous traffic patterns and identify potential DDoS attacks even with limited flow records.

3.1 Inter-Router Traffic Correlation: Observations from Real-World Networks

To validate our hypothesis that inter-router correlations provide key signals for volumetric DDoS detection, we investigated real-world NetFlow data from a major cloud service provider (CSP). This



(a) Pearson Correlation (b) Granger Causality

Figure 3: Inter-router correlation matrix: normal operations vs. DDoS attack conditions

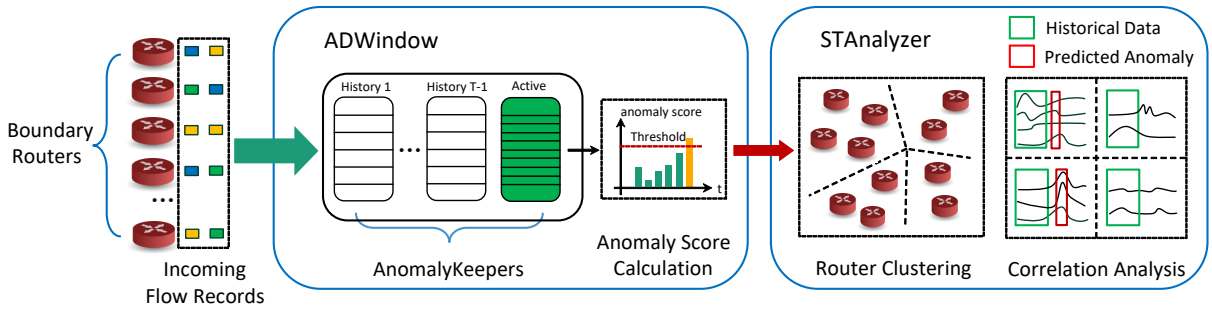


Figure 4: The overview of FlowSentry.

dataset contains traffic collected from multiple autonomous system (AS) domains across the country, with each flow record labeled as either normal or attack traffic based on verified security incidents. Detailed information about the dataset is provided in Section 6.1. From this extensive dataset, we selected 10 AS domains with the highest number of flows, treating the traffic data from each AS as representative of the flows observed by the corresponding boundary router. By further filtering flows that consistently traversed all of these AS domains, we focus on traffic patterns that span multiple boundary routers. We used normalized packet counts and byte counts from the NetFlow records as our primary features.

Figure 3 presents the coefficients between routers during periods of normal network operation and DDoS attack events. Each cell in the heatmap represents the correlation strength between two routers, with darker colors indicating stronger correlations. As shown in 3(a), during normal operation, most router correlations are moderate, with values concentrated below 0.55. It also reveals how network strategies shape these correlations. The intra-group correlations among routers (e.g., Routers 1–4, 5–7, and 8–10) are generally stronger than the inter-group correlations. During DDoS attacks, the coefficients between routers increase dramatically, with the average rising from 0.247 to 0.691. This substantial shift underscores the impact of DDoS attacks on inter-router traffic patterns, suggesting that enhanced inter-router correlations can serve as a key signal for attack detection. To further validate the temporal dependency aspects, we analyzed the causal relationships between router traffic patterns using Granger causality tests, as shown in Figure 3(b). The Granger causality results highlight significant changes in the directional influence between routers during DDoS attacks. Specifically, the network shows a stronger and more consistent causality during attack periods, with the causality coefficients increasing by approximately 109.5%. This aligns with distributed botnet behavior, where attackers coordinate multiple sources to maximize attack efficiency [15]. This coordination generates synchronized traffic patterns across multiple routers, creating the stronger causal dependencies we observe.

3.2 System Architecture

Based on the insights gained from the above empirical study, FlowSentry is designed to leverage the cross-router traffic correlations to accelerate DDoS attack detection. However, to effectively manage the extensive spatiotemporal data required for identifying network anomalies, FlowSentry faces a critical tradeoff between correlation analysis depth and detection timeliness. Conducting detailed correlation analysis across multiple routers introduces substantial computational and memory demands, which delays the DDoS detection. To address this challenge, FlowSentry builds on a dual-layer screening mechanism to achieve an efficient balance between the depth of analysis and the operational speed. In the initial layer, the basic anomaly scoring function (detailed in Section 4.4) is applied to quickly filter out the majority of benign traffic with moderate computational effort. This first layer reduces the volume of data passed to the more resource-intensive second layer, which conducts deeper analysis on flows flagged as potentially anomalous.

As shown in Fig. 4, FlowSentry integrates these strategies through its two core components, ADWindow and STAnalyzer.

Data flow in FlowSentry. The input to FlowSentry consists of NetFlow records collected from multiple boundary routers, which include flow features such as packet counts and byte counts, detailed in Table 1. FlowSentry incorporates only two additional features, historical flow records and router IP addresses, compared to industry-standard detection solutions. Both can be directly obtained from NetFlow records without incurring significant additional data collection overhead. These records are first processed by ADWindow, which stores the data and performs an initial screening to filter out the majority of benign flows based on the anomaly scores. Once potential anomalies are detected, ADWindow exports a spatiotemporal feature matrix derived from the flagged flows, containing their historical and current traffic information. STAnalyzer then uses this matrix for detailed correlation analysis to identify a list of target IP addresses under DDoS threat.

ADWindow. FlowSentry utilizes a sketch-based structure to compress historical data and facilitate high-frequency preliminary screening. Each sketch, referred to as an AnomalyKeeper, is dedicated to storing data over a specific period, managing the temporal dimensions of flow data. With each incoming flow record, AnomalyKeeper calculates its anomaly score based on the information in the current window. If the score exceeds a predefined threshold, the flow is flagged for further analysis.

STAnalyzer. STAnalyzer performs a more detailed analysis of the flagged data. During offline training, it first clusters routers into communities based on their flow patterns to reduce complexity and focus on segments prone to anomalies. Once the routers are clustered, STAnalyzer uses labeled historical data of cross-router large flows to jointly model both attack and background traffic patterns across multiple routers. During online detection, it predicts each flagged flow's traffic growth within each community and flags a flow as anomalous if its estimated traffic increment above background traffic exceeds a predefined threshold.

4 ADWindow

ADWindow is designed to keep track of potential DDoS attack flows and filter out benign flows to minimize the computational resources wasted on non-anomalous traffic. It operates as a high-frequency processing mechanism to handle massive volumes of flow records in near real-time, without waiting for the completion of an entire NetFlow reporting cycle.

4.1 Sliding Window Mechanism

To support both immediate detection and long-term pattern recognition, ADWindow maintains a hierarchical time window structure consisting of one active window and $T-1$ historical windows, where T denotes the window size. These windows operate at different temporal granularities: the active window captures recent flow dynamics with fine-grained time slices, while historical windows summarize earlier traffic patterns using coarser slices. Each time slice, whether in the active or historical window, is represented using an AnomalyKeeper sketch (detailed in Section 4.3) to compactly summarize flow features. When flow records arrive, their continuous features (e.g. byte count and packet count) are decomposed and mapped to appropriate time slices based on their timestamps in the active window, ensuring temporal alignment across varying reporting times and durations.

The historical windows correspond to a single reporting cycle τ. To ensure complete preservation of necessary flow records for the historical windows, the active window spans two reporting cycles (2τ), providing sufficient overlap for seamless transitions and comprehensive data capture. As the sliding window advances every τ, the expired fine-grained slices of active window are merged into coarser-grained slices in the historical window. This consolidation combines multiple AnomalyKeeper sketches, preserving key anomaly signals while reducing overhead. The multi-resolution design allows ADWindow to retain detailed short-term insights and efficiently capture long-term patterns.

The output of ADWindow is an $N \times ((T-1) \times t_{\text{history}} + t_{\text{active}})$ spatiotemporal feature matrix of potential anomalous flows, where N is the number of routers, and t_{history}, t_{active} denote the number of time slices per historical and active window, respectively.

4.2 Sketch Preliminary

Sketches are compact data structures designed to process streaming data by compressing large volumes of information into manageable summaries. This technique is particularly beneficial in environments where when storing all incoming data is impractical due to memory constraints [1, 39, 40]. Sketches use probabilistic data structures to efficiently estimate various network statistics. When a new data element arrives, multiple hash functions map it to specific positions in an array of buckets. In the classic CM Sketch [11], these buckets are incremented to reflect the element's presence. To estimate the frequency of an element, the same hash functions determine its positions in the arrays, and the minimum value among these buckets is taken as the estimated frequency. The accuracy of these estimates depends on the number of hash functions and the size of the arrays, which control the error margin.

4.3 The AnomalyKeeper Structure

As shown in Fig. 5, AnomalyKeeper consists of d arrays, each containing w buckets. Each bucket has three fields: a fingerprint field, an anomaly score field and a feature vector array field.

For clarity, we use $L_j[t]$ to represent the t-th bucket in the j-th array. Here, $L_j[t].FP$, $L_j[t].A$, and $L_j[t].Arr$ correspond respectively to the fingerprint, anomaly score, and the array of flow feature vectors. Each of the d arrays $L_1, \ldots, L_d$ is paired with a distinct hash function, and we have d independent hash functions $h_1(.), \ldots, h_d(.)$.

The fingerprint FP is obtained by taking the first 16 bits of the hash value generated by the hash function $h(.)$, while the anomaly score A is determined by the anomaly scoring function $AF(.)$ defined in section 4.4. The length of the array corresponds to the number of routers in the network. Our objective is to store the flow with the highest anomaly score within each bucket.

Insertion Process: Initially, all fingerprint fields are set to null, all anomaly scores are 0, and all arrays are empty. Upon the arrival of an incoming flow record R associated with flow f_i, AnomalyKeeper calculates the d hash functions to map f_i to d buckets $L_j[h_j(f_i)]$

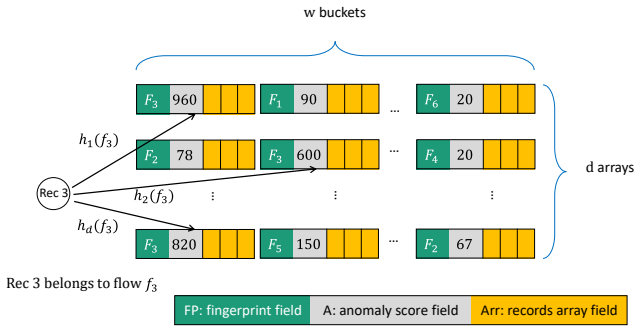


Figure 5: The data structure of AnomalyKeeper.

Table 2: Definitions of symbols used in AnomalyKeeper

Symbol	Description
L_j	The j-th array in AnomalyKeeper.
$h_j(\cdot)$	The hash function for the j-th array.
R	The incoming flow record.
f_i	The flow associated with the incoming records R.
F_i	The fingerprint of f_i in L_j.
A'	The anomaly score $L_j[h_j(f_i)].A$ for convenience.
A_R	The anomaly score calculated with R.
R.RouterID	The identifier of the router reporting the record R.

(where $1 \le j \le d$), termed as mapped buckets for simplicity. The symbols used in this process are defined in Table 2.

Case 1 (Empty Bucket): When $A' = 0$, it indicates that no prior flow is recorded in this bucket. AnomalyKeeper then initializes $L_j[h_j(f_i)].FP$ with the fingerprint F_i, sets A' to A_R, and populates $L_j[h_j(f_i)].Arr[R.routerID]$ with the feature vector extracted from R.

Case 2 (Existing FP Matches): If $A' > 0$ and $L_j[h_j(f_i)].FP = F_i$, it suggests that the current bucket already holds data for a flow with a matching fingerprint. Here, AnomalyKeeper increments A' by A_R, updates $L_j[h_j(f_i)].Arr[R.routerID]$ with feature vector extracted from R.

Case 3 (Existing FP conflicts): When $A' > 0$ and $L_j[h_j(f_i)].FP \neq F_i$, indicating that a different flow occupies the bucket, AnomalyKeeper applies an exponentially-weakening decay to A_R based on A'. The decayed anomaly score is computed as follows:

$$A' = A' - b^{-A'} * A_R$$

where b is a predefined exponential base number. This mechanism allows flows with low anomaly scores (benign flows) to be decayed quickly, while keeping those with high anomaly scores (malicious flows) in the bucket. If A' decays to zero, AnomalyKeeper replaces $L_j[h_j(f_i)].FP$ with F_i, resets A' to A_R, and updates $L_j[h_j(f_i)].Arr$ accordingly.

Query Process: To query the anomaly score of a flow f_i, AnomalyKeeper computes the d hash functions to identify d buckets $L_j[h_j(f_i)]$ (where $1 \le j \le d$). Among these buckets, it identifies those where the fingerprint fields match F_i and reports the maximum anomaly score from these buckets:

$$\max_{1 \le j \le d} \{L_j[h_j(f_i)].A\} \quad \text{where} \quad L_j[h_j(f_i)].FP = F_i.$$

Merging Process: When transitioning time slices from the active window to historical windows, AnomalyKeeper instances need to be merged to maintain coarse-grained temporal representations. Given two AnomalyKeeper instances $AK1$ and $AK2$ that need to be merged, the process examines each corresponding bucket pair at the same position:

Case 1 (Empty Bucket): If a bucket exists in only one AnomalyKeeper (i.e., the corresponding bucket in the other is empty), the merged AnomalyKeeper directly inherits the non-empty bucket.

Case 2 (FP matches): When $AK1.L_j[k].FP = AK2.L_j[k].FP$, the merged AnomalyKeeper adds their feature vectors element-wise and recalculates the anomaly score based on the combined result.

Case 3 (FP conflicts): When $AK1.L_j[k].FP \neq AK2.L_j[k].FP$, the merged AnomalyKeeper retains the fingerprint and features from the bucket with the higher anomaly score.

For simplicity, if $L_j[h_j(f_i)].FP = F_i$, we say that f_i is held in the bucket $L_j[h_j(f_i)]$. Despite potential fingerprint collisions, we ensure that the reported anomaly score reflects an accurate assessment of the flow's deviation from the typical network behavior. If a flow is not held in any mapped bucket, it is reported as benign. Conversely, if a flow is held in multiple buckets, AnomalyKeeper reports the highest anomaly score to accurately capture significant deviations.

4.4 Anomaly Scoring Function

Given the high-frequency data processing requirements of AnomalyKeeper, the anomaly scoring function $AF(\cdot)$ is designed to be computationally efficient while still providing a reliable indication of potential threats.

Features. The anomaly scoring process relies on three fundamental features collected from all time slices within the most recent reporting period to form a temporal feature vector:

- R_b: Byte rate
- R_p: Packet rate
- C: Number of routers the flow passes through

To capture sudden anomalies, we introduce additional features representing the most recent changes of these three metrics. Each delta feature (ΔR_b, ΔR_p, ΔC) is calculated as the difference between the current and immediately previous time slice. Therefore, the final feature vector X for anomaly scoring is constructed as follows:

$$X = [\mathbf{R_p}, \mathbf{R_b}, \mathbf{C}, \Delta R_p, \Delta R_b, \Delta C]$$

where $\mathbf{R_p}$, $\mathbf{R_b}$, and $\mathbf{C}$ represent the temporal sequences of packet rate, byte rate, and router count across all time slices in the recent reporting period, respectively.

Anomaly Scoring Function Models. Various models can be used to compute the anomaly score A based on these features, covering models of varying complexity. For faster detection, a linear model can be employed, while for more precise detection, an SVM

model is a suitable choice. Based on our experiments (detailed in Section 6.5), we recommend using the moderate option of K-means to trade off between delay and precision. However, users are free to define their own functions as needed.

Linear Model. For computational efficiency, the linear model uses only statistical summaries of the temporal features. For each fundamental feature (R_p, R_b, C), we calculate the minimum, maximum, mean, and standard deviation across all recent time slices, along with the delta features. The linear function is then expressed as:

$$A = \mathbf{w}^T \mathbf{S}$$

where $\mathbf{S}$ is the statistical feature vector:

$$\begin{aligned}\mathbf{S} = [\ &\min(R_p), \max(R_p), \text{mean}(R_p), \text{std}(R_p), \Delta R_p, \\ &\min(R_b), \max(R_b), \text{mean}(R_b), \text{std}(R_b), \Delta R_b, \\ &\min(C), \max(C), \text{mean}(C), \text{std}(C), \Delta C]\end{aligned}$$

and $\mathbf{w}$ is the corresponding weight vector.

K-means Clustering Model. The K-means function calculates the anomaly score as the minimum distance between the feature vector and the nearest cluster centroid:

$$A = \min_k \text{distance}(X, \mu_k)$$

where μ_k denotes the centroid of each cluster.

SVM-Based Model. The SVM-based function employs a Gaussian (RBF) kernel and calculates the anomaly score based on the distance from the decision hyperplane, which serves as a boundary separating different classes within the feature space:

$$A = \text{distance}(X, \text{hyperplane})$$

Pre-training Phase. Model parameters and detection thresholds are determined using historical network data. The linear model weights $\mathbf{w}$ are learned through supervised training on labeled data. K-means centroids μ_k are obtained via unsupervised clustering of normal traffic patterns. SVM hyperplane parameters are trained using supervised learning with the RBF kernel. Detection thresholds for all models are set by analyzing score distributions on validation data to balance false positive and false negative rates.

5 STAnalyzer

In this section, we design a spatio-temporal correlation algorithm to deeply detect potential anomalous flows marked by ADWindow, as outlined in Alg. 1.

5.1 Rationale

STAnalyzer leverages multi-router traffic correlation to efficiently infer abnormal growth patterns of the partially reported flow records. By analyzing both spatial (multiple routers) and temporal (historical traffic data) dimensions, STAnalyzer enhances detection capabilities single-router analysis.

To further enhance detection efficiency, STAnalyzer leverages the observed regional correlation of attack sources by clustering routers into communities based on historical traffic patterns. Attack sources typically exhibit coordinated behaviors within regions and tend to remain stable over short periods, as frequent migration is costly for attackers. This clustering method reduces analysis complexity by focusing on network segments where anomalies are more likely to occur or have the greatest impact.

Within each router community, STAnalyzer applies a Vector Autoregression (VAR) model to capture interdependencies among community members using labeled historical data from large flows that traverse multiple routers. To capture the deviation of target flows from background traffic, we employ a joint optimization strategy that simultaneously models both types of traffic. An asymmetric penalty is applied during training to further enhance detection sensitivity and robustness, encouraging aggressive prediction of attack traffic growth while maintaining conservative estimation of normal traffic.

During online inference, STAnalyzer predicts both the target flow and background traffic for each traversed community, and flags a flow as anomalous if its aggregated traffic increase over the background exceeds a predefined threshold. To adapt to evolving traffic patterns, VAR models are updated via both online incremental updates with recently observed cross-router elephant flows and periodic offline full retraining using labeled historical data.

Algorithm 1 STAnalyzer

Input: Alerted IPs, historical labeled data, real-time data stream
Output: Detected anomalies
Initialize threshold θ and VAR lag length p
Detect communities from historical traffic patterns
for each community com_i **do**
    Train VAR_i with labeled data from com_i
while ADWindow reports suspicious flow **do**
    $traffic\_increment \leftarrow 0$
    for each com_i traversed by the flow **do**
        $sus\_traffic \leftarrow VAR_i$(suspicious flow data)
        $bg\_traffic \leftarrow VAR_i$(background data)
        $traffic\_increment \leftarrow sus\_traffic - bg\_traffic$
    Report anomaly if $traffic\_increment > \theta$

5.2 Router Clustering

To accelerate the detection process, routers are first clustered into communities. To achieve this, we employ community detection algorithms from graph theory.

Graph Construction: Each router is represented as a node, and the weight of the edge between nodes represents the correlation coefficient between the routers based on flows with the same destination IP address. This coefficient is calculated using the feature vectors stored by ADWindow. The corresponding formula is given by:

$$\text{weight}(i,j) = \frac{\sum_{\text{flow}}(f_{i,\text{flow}} - \overline{f}_i)(f_{j,\text{flow}} - \overline{f}_j)}{\sqrt{\sum_{\text{flow}}(f_{i,\text{flow}} - \overline{f}_i)^2}\sqrt{\sum_{\text{flow}}(f_{j,\text{flow}} - \overline{f}_j)^2}}$$

where $f_{i,\text{flow}}$ and $f_{j,\text{flow}}$ are the feature vectors of the same flow *flow* at routers i and j, and $\overline{f}_i$ and $\overline{f}_j$ are the mean feature vectors of flows at routers i and j over all matching flows.

Community Detection Algorithm: The Louvain algorithm is exploited to analyze the constructed graph by grouping routers into distinct communities. This method aims to maximize the modularity

Q, which quantifies the density of connections within communities compared to those between communities. The modularity is calculated as follows:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

where A_{ij} is the weight of the edge between nodes i and j, k_i and k_j are the sum of the weights of the edges attached to nodes i and j respectively, m is the sum of all edge weights in the graph, and $\delta(c_i, c_j)$ is 1 if nodes i and j are grouped into the same community and 0 otherwise.

The algorithm starts by treating each node as an individual community and then progressively merges communities to maximize the modularity Q. The number of communities to be formed can be adjusted through the resolution parameter. This process repeats until no further improvement in modularity can be achieved.

5.3 Community-based Traffic Prediction

For each pre-trained router community, we use the VAR model to capture the linear interdependencies among multiple time series.

Vector Autoregression (VAR) Modeling. Assume a community contains k routers (i.e., router 1, router 2, ..., router k). For router i, let $f(i,t)$ denote the flow feature vector at time t, consisting of two components: $f(i,t).r_p$ for packet rate and $f(i,t).r_b$ for byte rate. Due to the discrete nature of flow reporting, features remain constant within each reporting interval T. In other words, if t_i is the last reporting time for router i, then $f(i,t) = f(i,t_i)$ for $t_i \le t < t_i + T$. Let F_t be the matrix representing the flow features of all routers at time t, defined as:

$$F_t = \begin{pmatrix} f(1,t) & f(2,t) & \cdots & f(k,t) \end{pmatrix}^\top$$

The lag length is denoted as p that identifies how many past time steps to use for predictions. Upon receiving the real-time flow records at time t, we aim to predict the **maximum flow features** $\hat{F}^{\max}_{t,t+T}$ over the next reporting interval $[t, t+T]$:

$$\hat{F}^{\max}_{t,t+T} = c + A_1 F_t + A_2 F_{t-1} + \cdots + A_p F_{t-p+1} + \epsilon_{t+T}$$

where c is a vector of constants, A_i are coefficient matrices, and ϵ_{t+T} is a matrix of error terms.

Joint Optimization. The training process involves estimating the parameters $c, A_1, A_2, ..., A_p$ by jointly minimizing the losses for the target flow and background traffic:

$$\min \sum_{t \in \mathcal{T}} \left[L_s \left(F^{s,\max}_{t,t+T}, \hat{F}^{s,\max}_{t,t+T} \right) + \lambda \left\| F^{bg,\max}_{t,t+T} - \hat{F}^{bg,\max}_{t,t+T} \right\|_F^2 + \zeta \sum_{i=1}^{p} \|A_i\|_F^2 \right]$$

where $F^{s,\max}_{t,t+T}$ and $F^{bg,\max}_{t,t+T}$ denote the maximum feature matrices of the target flow and average background traffic within the interval $[t, t+T]$. λ and ζ are weighting parameters, $\|\cdot\|_F$ denotes the Frobenius norm, and $\mathcal{T}$ is the set of training time points.

An asymmetric penalty is applied in the target flow loss function:

$$L_s = w_u(\text{label}) \left\| \max\left(0, F^{s,\max}_{t,t+T} - \hat{F}^{s,\max}_{t,t+T}\right) \right\|_F^2 + \left\| \max\left(0, \hat{F}^{s,\max}_{t,t+T} - F^{s,\max}_{t,t+T}\right) \right\|_F^2$$

where weights are set based on flow labels:

$$w_u(\text{label}) = \begin{cases} 1+\alpha & \text{if label = attack} \\ 1-\alpha & \text{if label = normal} \end{cases}$$

Here, $0 < \alpha < 1$ is a hyperparameter that controls the strength of asymmetric penalty. This asymmetric training strategy ensures the model is sensitive to attack growth while remaining conservative about normal traffic variations.

Real-Time Anomaly Detection. When ADWindow reports a suspicious flow at time t, STAnalyzer applies the VAR model to predict the next reporting period's traffic F_{t+T}. The flow is flagged as anomalous if:

$$\sum_{j=1}^{C} \sum_{i \in \text{com}_j} \left(\hat{f}^s(i, t+T).r_b - \hat{f}^{bg}(i, t+T).r_b \right) > \theta$$

where C is the number of communities traversed by the suspicious flow, com_j denotes the j-th community, and θ is a predefined threshold.

Adaptive Model Update. STAnalyzer employs two update mechanisms to maintain prediction accuracy under evolving traffic conditions. *Online incremental updates* adapt to short-term traffic variations by minimizing prediction errors on flows that traverse multiple routers with sufficient volume, treating them as normal traffic. *Offline full retraining* occurs at longer intervals using labeled data, applying asymmetric optimization to preserve attack detection sensitivity while incorporating recent traffic patterns.

6 Evaluation

6.1 Experiment Setup

For the evaluation of FlowSentry, we simulate a NetFlow detection system where 100 boundary routers sent NetFlow records to a central detector. The NetFlow reporting interval was set to 60 seconds, which is a standard reporting time in the industry. The initial reporting time for each router was randomly distributed within the first 60 seconds.

Dataset. We utilized labeled traces from three sources: synthesized NetFlow from public packet-level data (CIC-DDoS dataset), real-world traffic from a Tier-1 ISP (ISP dataset), and AS-annotated flow data from a major cloud service provider (CSP dataset).

- **CIC-DDoS dataset.** Due to the lack of publicly available NetFlow datasets from backbone networks, we utilized fine-grained packet-level data to synthesize realistic NetFlow records. As background traffic, we employed daily traces from samplepoint-F of MAWI [37] from February to June 2023, which monitors a transit link connecting to a Tier 1 ISP in Japan. For the attack traffic, we utilized five types of DDoS attack traces from the CIC-DDoS-2019 dataset [31] which is widely used for evaluating DDoS detection algorithms. The attack traffic included SYN flood, UDP flood, SSDP reflection, DNS reflection and NTP reflection attacks. To ensure the traffic scale aligns with backbone network environments, we applied differential sampling rates when converting packet-level data to NetFlow records. The specific sampling rates were determined based on the respective traffic statistics (attack vs. normal) observed in the Tier-1 ISP dataset, implemented using standard NetFlow tools (i.e., nfdump and softflowd).
- **ISP dataset.** This dataset was provided by a Tier 1 Internet service provider and consists of NetFlow records collected from 8 boundary routers deployed at the ingress points of

Table 3: Detection accuracy and delay for FlowSentry and baseline methods.

Dataset	Method	S. T. Detector				Jaqen				Whisper				FlowSentry			
	Metric	AUC	F1 Score	FPR	Delay (s)	AUC	F1 Score	FPR	Delay (s)	AUC	F1 Score	FPR	Delay (s)	AUC	F1 Score	FPR	Delay (s)
CIC-DDoS Dataset	SYN Flood	0.903	0.858	0.021	93.67	0.923	0.902	0.028	59.21	0.905	0.885	0.036	66.82	0.927	0.907	0.016	20.14
	UDP Flood	0.921	0.873	0.019	91.13	0.917	0.876	0.033	57.48	0.903	0.892	0.027	60.32	0.951	0.931	0.019	14.98
	SSDP Reflection	0.852	0.807	0.022	95.15	0.841	0.803	0.031	69.12	0.887	0.854	0.026	70.01	0.906	0.862	0.019	22.50
	DNS Reflection	0.936	0.912	0.023	96.22	0.929	0.904	0.052	56.34	0.923	0.893	0.041	66.86	0.941	0.917	0.020	18.06
	NTP Reflection	0.890	0.843	0.031	92.39	0.890	0.839	0.033	58.33	0.929	0.901	0.029	63.80	0.927	0.904	0.037	19.11
ISP Dataset	RST Flood	0.909	0.895	0.033	86.73	0.915	0.897	0.036	55.86	0.919	0.889	0.044	64.27	0.921	0.901	0.031	21.75
	UDP Flood	0.925	0.912	0.024	90.34	0.921	0.901	0.031	56.90	0.931	0.918	0.029	61.82	0.933	0.921	0.022	18.59
	DNS Reflection	0.935	0.921	0.024	89.52	0.930	0.910	0.028	54.93	0.932	0.916	0.026	63.15	0.934	0.929	0.021	24.32
	Memcached Reflection	0.917	0.913	0.021	93.17	0.919	0.901	0.033	59.52	0.902	0.886	0.063	72.13	0.941	0.933	0.018	20.76
CSP Dataset	SYN Flood	0.942	0.913	0.024	87.45	0.951	0.919	0.027	54.87	0.934	0.911	0.040	64.16	0.940	0.915	0.030	19.85
	ACK Flood	0.928	0.902	0.031	88.14	0.919	0.899	0.051	58.33	0.911	0.892	0.045	63.47	0.937	0.912	0.026	16.14
	UDP Flood	0.932	0.908	0.027	86.91	0.939	0.913	0.030	55.96	0.937	0.915	0.045	67.31	0.942	0.921	0.021	21.23

[1] Blue cells indicate the highest AUC values, orange cells denote the best F1 scores, red cells highlight the lowest FPR values and green cells mark the lowest detection delays.

provincial networks, which are located downstream of the national backbone. Each flow record is annotated by the provider to indicate the originating backbone-level boundary router. Based on these annotations, we partitioned the flow records according to their associated backbone-level boundary routers and selected the 100 routers with the highest traffic volumes for analysis. The flow data was sampled at a high rate of 2000:1. On average, the dataset captures approximately 0.54 million flow records per second, with an aggregate packet rate of 1.12 million packets per second (Mpps). Approximately 0.4% of the traffic is labeled as DDoS attack flows. The attacks last between 2 and 8 minutes, with peak intensities ranging from 1.23 Mpps to 9.44 Mpps.

- **CSP dataset.** This dataset was collected by a major cloud service provider and contains two types of data annotated with AS attribution labels: (i) flow records, and (ii) DDoS command-and-control (C2) information obtained from botnet servers captured by nationwide honeypots. The C2 data includes key attack parameters such as initiation time, duration, and traffic rate. As described in Section 3.1, we used the AS labels to partition the flow records into router-level traffic views. For analysis, we selected the 100 AS domains with the highest traffic volumes. The dataset includes labeled DDoS attack traffic with durations ranging from 1 to 7 minutes, and peak intensities ranging from 36.4 Kpps to 171.6 Kpps.

Multi-Router Traffic Emulation. To adapt the CIC-DDoS dataset to our multi-router experimental setting, we emulated traffic distribution across multiple boundary routers as follows: (i) For benign traffic, we used MaxMind's GeoIP database [28] to map source IP addresses to AS domains and assigned flows to simulated routers accordingly. (ii) For attack traffic, we used the AS labels from the command scripts in the CSP dataset to assign attack flows to 80 simulated boundary routers (80% of all routers). Each assigned router then replayed CIC-DDoS attack flows according to the corresponding script, creating distributed attack scenarios with realistic spatial and temporal characteristics.

Training and Testing Data Split. For each dataset, we divided the traffic into training and testing sets in chronological order, with approximately 70% of the traffic allocated for training and the remaining 30% for testing. The training dataset was derived from earlier timestamps and included both benign traffic and samples of DDoS attack traffic to establish foundational patterns. The testing dataset was constructed independently with different DDoS attack instances temporally separated from those in the training data. This strict separation prevented data leakage and ensured test attack patterns were not continuations of training attacks. To standardize our testing environment and allow sufficient time for method initialization, all attack occurrences in the testing dataset were configured to begin after the 300th second of each test scenario.

Baseline. We select three detection methods as baselines, covering industry-standard threshold-based detection and two recent research proposals. For each method, we adopt the hyperparameters that follow industry best practices or are recommended in their respective papers.

- **Simple Threshold Detector.** This detector periodically maintains the packet rate and byte rate of each flow during the detection period. Once it detects rates exceeding the threshold, which is set based on the optimal results from a training dataset, it considers an attack to have occurred. The detection period is an integer multiple of the NetFlow reporting interval. At the end of each detection period, the maintained flow record cache is cleared and restarted. The detector, provided by the security department of a Tier 1 ISP, is widely adopted in the industry due to its simplicity and effectiveness in detecting malicious traffic with minimal computational overhead. In the accompanying figures and tables, this method is abbreviated as S. T. Detector.
- **Jaqen.** Jaqen [27] employs a universal sketch to compute flow statistics and applies a multi-threshold detection approach. This method has been extended to incorporate NetFlow statistical features, with the corresponding thresholds adjusted based on the training dataset to better distinguish between normal and malicious traffic. Other operations remain the same as the simple threshold detector, including the periodic reset of the flow record cache. Although Jaqen is designed for distributed deployment across multiple programmable devices, we focus solely on server-side optimization in our evaluation and therefore use only the single-node version of Jaqen for comparison.
- **Whisper.** Whisper [13] encodes each packet's length, inter-packet delay, and type into a floating-point value, followed by applying a Fourier transform to the sequence of packet features. It then utilizes an unsupervised clustering method to identify victim hosts. To ensure a fair comparison, we

reproduced a flow-based version of Whisper by replacing packet-level features with flow-level features and employing a supervised decision tree model for detection.

We exclude other methods for two main reasons: methods involving router modifications are not practical for real-world deployment, and deep learning-based approaches are too computationally intensive to meet our low-latency NetFlow processing objective.

Metric Selection. We utilize both AUC (Area Under the ROC Curve) and F1 score to evaluate the accuracy of our model, as they are widely used in [13, 14, 29]. Additionally, we include the False Positive Rate (FPR) to measure the proportion of benign flows that are mistakenly identified as attacks, which is critical for avoiding disruption to legitimate online services. Delay, defined as the time elapsed from the occurrence of an attack to its successful detection, is used to measure detection responsiveness. Note that we exclude undetected attack flows from delay calculations, as detection failures are already accounted for by precision-related metrics.

Implementation. Our experiments are run on a server with dual 18-core Intel Xeon Gold 6240C CPUs (72 threads, 2.60 GHz) and 251 GiB. All detectors are implemented in C++. We set $d = 4$ for each AnomalyKeeper in FlowSentry, and the window size is set to 5. The fingerprint field and the anomaly score field are both 16-bit long, and each feature vector array field is 16-bit×3 long. The number of buckets is set to 3000.

6.2 End-to-End Performance

Table 3 summarizes the performance of FlowSentry in comparison to baseline methods. On average, FlowSentry achieves an improvement of 0.65% in AUC and 0.92% in F1 score, while reducing the FPR by 6.44% compared to the best-performing baselines for each attack scenario. More significantly, FlowSentry exhibits a dramatic reduction in detection delay by 65.63%, which reflects its strength in enabling timely detection.

In terms of accuracy, FlowSentry achieves the highest AUC and F1 scores in most attack scenarios, along with a notably lower FPR than competing methods. This is because FlowSentry effectively leverages cross-router correlations to identify anomalous traffic patterns in a more comprehensive way, even when using partially reported flow data. Nevertheless, in a few individual cases such as the NTP Reflection in the CIC-DDoS dataset, FlowSentry's AUC is marginally lower (0.927 vs. 0.929) than that of Whisper. This slight drop in accuracy can be attributed to FlowSentry's more aggressive strategy in predicting traffic growth trends, which prioritizes early detection at the occasional cost of detection precision. In contrast, the simple threshold detector adopts a more conservative approach, only identifying anomalies once the sum of rates at all points definitively exceeds the threshold.

In terms of delay, FlowSentry exhibits a substantial improvement over all baseline methods by reducing detection time to an average of 19.46 seconds. Specifically, FlowSentry's delay ranges from 14.98 seconds to 24.32 seconds across various datasets and attack types, which is about one-third of a reporting interval. That being said, FlowSentry is able to detect attacks using approximately one-third of the flow records after they occur. In contrast, Jaqen and Whisper typically detect attacks in about one reporting interval, requiring the complete set of flow records reported by all routers after the attack. The simple threshold detector requires approximately half of the routers to report flow records twice after the attack begins, typically detecting attacks in about one and a half reporting intervals. The result shows that FlowSentry manages to balance high detection accuracy with rapid detection, ensuring timely mitigation of volumetric DDoS attacks.

FlowSentry also demonstrates robust generalizability across heterogeneous network environments, as evidenced by its consistent performance on all three datasets. The three datasets span diverse environments, including fine-grained enterprise traffic from the CIC-DDoS dataset, geographically and topologically diverse AS-level attacks from the CSP dataset, and coarse-grained backbone-level flows from the ISP dataset. On average, FlowSentry reduces detection delay by FlowSentry reduces detection delay by 68.51%, 62.31%, and 66.07% on the CIC-DDoS, ISP, and CSP datasets respectively, while maintaining superior detection accuracy. This cross-environment robustness is critical for real-world deployment, where traffic sources, sampling strategies, and router coverage can vary significantly.

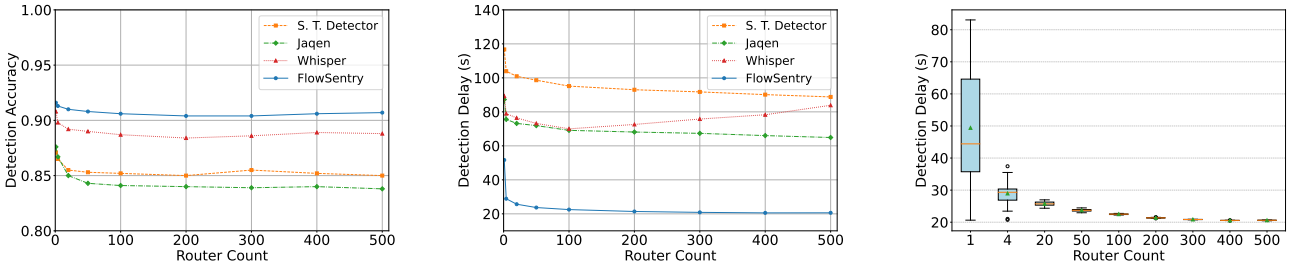
6.3 Impact of Router Scale

We conducted controlled experiments to explore the impact of router scale on detection performance. Specifically, we maintained constant parameters, including the total number of flows and the scale of each flow. Additionally, for each concurrent flow traversing multiple routers, we ensured that it passed through a similar proportion of routers as the number of routers increased.

Fig. 6(a) and 6(b) demonstrate the changes in detection accuracy and delay of baseline methods under the SSDP Reflection attack scenario. When the number of routers is reduced to one, the detection falls back to a single-point detection scenario, resulting in coarser detection granularity. This coarser granularity leads to an increase in both detection accuracy and detection delay, as more flow records accumulate during the detection process. For example, if a detector requires flow records spanning 1.5 reporting intervals to identify an attack, the detection will succeed in the second interval if the attack begins in the first half of the initial interval, or in the third interval if it starts in the latter half. As a result, the coarse granularity of reporting leads to the accumulation of redundant flow records beyond what is minimally required for an accurate detection by the time detection is achieved. This redundancy contributes to an increase in detection accuracy while simultaneously introducing higher detection latency.

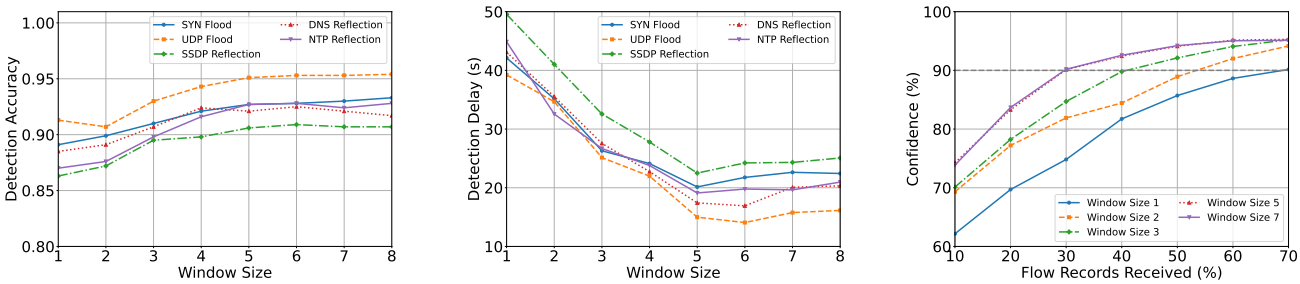
As the scale of routers increases, the detectors benefit from finer-grained and higher-frequency flow record reporting. This allows DDoS detection to occur closer to the moment when sufficient flow records become available, thereby reducing detection latency. Although higher reporting frequency inevitably increases computational overhead due to the greater volume of flow records to process, FlowSentry maintains scalability because the majority of additional records are are filtered out by the lightweight ADWindow. As a result, FlowSentry achieves a similar level of delay reduction as other high-speed detection methods(i.e., Simple Threshold Detector and Jaqen), whereas Whisper fails to maintain comparable performance.

Fig. 6(c) illustrates the delay distribution of FlowSentry across varying router counts. With a single router, the delay distribution exhibits high variance, with a wide interquartile range and several



(a) AUC (b) Delay (c) Delay Distribution

Figure 6: Impact of router scale on FlowSentry and all the baselines.



(a) AUC (b) Delay (c) Confidence

Figure 7: Impact of window size on FlowSentry's performance.

outliers. This variance arises from the timing of obtaining the required flow records, which depends on the interval between two consecutive flow record reports. With only one router, the coarse time granularity between these two reporting moments introduces significant variability in detection delay. In contrast, as the number of routers increases, the time granularity improves, and the interval becomes shorter. This reduction in variability significantly decreases the delay variance and allows the detection delay to stabilize around a lower median value. Consequently, in large-scale router deployments, the delay distribution becomes more predictable and consistent, ensuring reliable detection performance.

6.4 Impact of Window Size

We evaluated the impact of window size on FlowSentry's performance. As depicted in Fig. 7(a) and 7(b), using only the current reporting interval (window size = 1) results in the lowest detection accuracy and highest detection delay. As the window size increases, both detection accuracy and the speed of detection improve, reaching a more stable level after a window size of 5. This trend indicates that more historical information allows FlowSentry to better capture and analyze traffic patterns, detect anomalies more quickly by providing a richer context for identifying suspicious patterns.

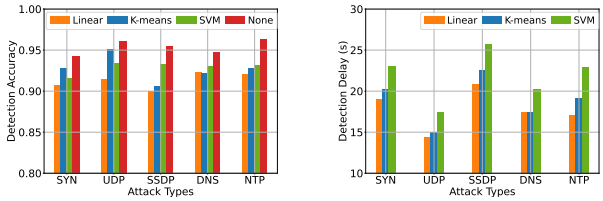
Fig .7(c) further validates this trend. The confidence score is computed as the probability that the traffic increase exceeds the decision threshold, based on the predicted increase and the empirical distribution of historical prediction errors. As the window size increases, FlowSentry's confidence in detecting anomalies improves significantly. This is because a larger window size provides richer historical context, enabling FlowSentry to better differentiate between normal and abnormal patterns in traffic. For example, when the window size is small (e.g., 1 or 2), FlowSentry requires more flow records to achieve a similar level of confidence compared to larger window sizes (e.g., 5 or 7). This highlights the benefit of incorporating more historical information, as it reduces the amount of data needed in real-time to reach high confidence levels.

However, adding more data beyond a certain window size yields diminishing returns. This occurs because networks constantly evolve, and outdated information not only fails to aid detection but also increases the processing time. We select a window size of 5 to capture sufficient historical context while avoiding outdated information.

6.5 Impact of Anomaly Scoring Function

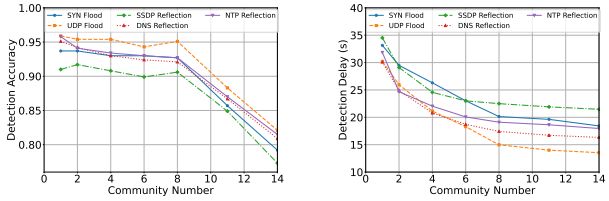
Fig. 8(a) illustrates the impact of different anomaly scoring functions on FlowSentry's accuracy, including a baseline condition where no initial screening was used (marked as "None"). All functions achieve accuracies above 0.90, demonstrating the effectiveness of FlowSentry's dual-layer detection architecture. The accuracy primarily depends on the fine-screening stage. The influence of the initial screening on accuracy, though less critical, still impacts the system's precision. The Linear function, due to its simpler design, has the lowest accuracy. The K-means and SVM functions demonstrate distinct advantages, accurately detecting various attack types.

Fig. 8(b) illustrates the impact on the detection delay. We omitted the delay data for the "None" scenario because it was too high to support real-time detection in our test environment. The SVM function exhibits the highest delays, primarily due to the extensive



(a) AUC (b) Delay

Figure 8: Impact of Anomaly Scoring Function on FlowSentry's performance.



(a) AUC (b) Delay

Figure 9: Impact of router clustering on FlowSentry's performance.

computation required by its kernel functions. Linear exhibits the shortest delays among the three, attributed to its simpler computational model. K-means closely follows Linear, indicating that while slightly slower, it offers a good compromise between speed and detection capability.

These observations highlight the trade-off between detection speed and accuracy introduced by different anomaly scoring functions. Based on these results, we provide the following guideline for selecting an appropriate scoring function tailored to specific deployment requirements. In latency-sensitive environments where fast response is critical, lightweight functions such as the Linear model are more suitable due to their low computational overhead. For scenarios where detection precision is prioritized over timeliness—such as low-rate attacks that trade bandwidth for stealth—SVM achieves higher accuracy, albeit with increased delay. K-means serves as a balanced choice, offering a practical compromise between detection effectiveness and computational efficiency, making it ideal for general-purpose use.

6.6 Impact of Router Clustering

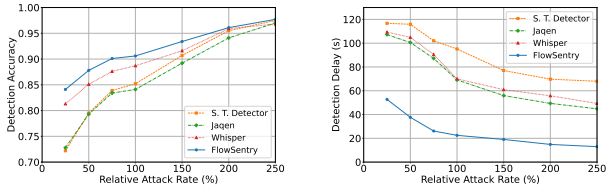
We adjust the resolution parameter of the Louvain algorithm to explore the impact of router grouping granularity (i.e., the number of communities). As shown in Fig. 9(a) and 9(b), when the community number is 1, meaning no router clustering is applied, both the accuracy and delay reach their maximum levels. In this scenario, all routers' interconnections are learned, which boosts detection accuracy while also extending the response time. As the community number increases, we observe a clear trade-off pattern. The reduction in parameters involved in learning correlations within each community leads to faster processing times. However, this reduction also causes a drop in detection accuracy due to the model's inability to capture interactions between routers across different communities. This limitation becomes particularly pronounced when communities are overly fragmented. Based on our experimental results, we selected a community number of 8 to balance the detection speed and the depth of analysis.

7 Evasion Risk Discussion

In this section, we examine the robustness of FlowSentry against evasion strategies [13, 30] that attackers may adopt to reduce detectability. Specifically, we investigate two categories of evasion risks: (i) Traffic rate throttling: attackers reduce the transmission rates of malicious traffic to try to stay below detection threshold; (ii) Cross-router correlation breaking: attackers manipulate the distribution of attack flow features to undermine FlowSentry's correlation-based detection assumption. To this end, we conduct controlled experiments in which the detection model is trained on the original dataset, while the test set is crafted with modified attack behaviors to simulate evasion scenarios.

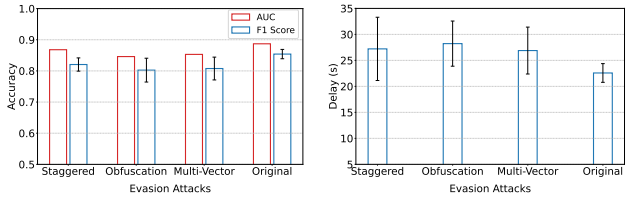
Traffic Rate Throttling. As shown in Figure 10, as attackers reduce the rate of malicious traffic, detection becomes more difficult due to weakened anomaly signals. While all methods experience degraded performance under such conditions, FlowSentry exhibits smaller reductions in both accuracy and responsiveness compared to baseline methods. More precisely, FlowSentry shows the smallest average AUC reduction of 3.61%, followed by Whisper (4.55%), Jaqen (6.66%), and S. T. Detector (7.82%). Despite this, FlowSentry maintains significantly lower delay under moderately reduced attack rates. For example, at 75% of the standard rate, its delay increases by only 3.6 seconds—49.4% less than the increase seen in the best-performing baseline. Under severely throttled attack conditions, FlowSentry's performance declines more noticeably. However, such extreme rate reduction is typically not cost-effective for attackers, as it leads to inefficient use of the botnet resources and reduces the likelihood of producing the rapid traffic surge needed for volumetric DDoS attacks. We also consider the scenario where attackers increase the attack rate in order to more rapidly overwhelm the bottleneck link. In such high-intensity attacks, FlowSentry benefits from clearer spatiotemporal patterns, enabling faster and more accurate detection. Its AUC rises to 0.977 and detection delay drops to 12.97 seconds at 250% of the standard rate.

Cross-router Correlation Breaking. We construct three adversarial variants: **(i) Staggered temporal pattern:** attackers distribute their malicious traffic across different time intervals, limiting the number of routers where correlated traffic patterns can be observed simultaneously. Specifically, attackers divide the routers into two groups and alternate their participation across consecutive reporting periods. In each period, one group sends attack traffic at 1.5× the standard rate, while the other reduces its rate to 0.5×. **(ii) Obfuscation:** Attackers mix legitimate flows with malicious traffic at a 1:4 ratio to obscure correlation signals and reduce anomaly visibility. **(iii) Multi-vector attacks:** attackers launch different types of DDoS traffic from different sources to reduce spatial correlation. In particular, 40% of the routers launch SSDP reflection attacks, while the remaining 60% are evenly split across SYN flood, UDP flood, DNS reflection, and NTP reflection attacks. As shown in Figure 11, FlowSentry remains effective across all three correlation-breaking strategies. The AUC decreases by 2.1%, 4.6%, and 3.8% under the



(a) AUC (b) Delay

Figure 10: Detection performance of FlowSentry and all the baselines under varying attack rates.



(a) Accuracy (b) Delay

Figure 11: Detection performance of FlowSentry under different evasion strategies.

staggered temporal pattern, obfuscation, and multi-vector attacks, respectively, while the F1 score drops by 1.7%, 7.3%, and 6.0%. The corresponding detection delays increase by 4.0, 5.1, and 4.7 seconds, but remain consistently below 30 seconds. We also observe an increase in the standard deviation of F1 scores and delays across different parameterizations of each evasion strategy, suggesting that evasion strategies introduce greater variability in detection outcomes. Nevertheless, the overall performance degradation remains marginal, and FlowSentry continues to provide accurate and timely detection in the presence of sophisticated evasive behaviors.

The limited effectiveness of these evasion strategies stems from the fundamental nature of volumetric DDoS attacks, which rely on the simultaneous transmission of large-scale traffic from distributed sources to rapidly saturate target network links. The inherent need for coordination in DDoS attacks naturally gives rise to temporal and spatial patterns as malicious traffic converges through multiple boundary routers. While evasion tactics may partially obscure these patterns, they cannot fully eliminate the systemic correlations that arise from the attack's synchronized structure. In addition, attempts to evade detection through fine-grained manipulation could incur significant overhead. Attackers must carefully control the behavior of distributed bots (e.g. manipulate timing and flow distribution), which increases the complexity and cost of maintaining an effective evasion campaign [15].

8 Related Work

Single-Point Detection Methods. Whisper [13] utilizes frequency domain attributes for real-time detection. HyperVision [14] detects encrypted attacks by analyzing flow interaction patterns through graph structural features. Flowlens [5] identifies attacks by extracting flow distribution characteristics on the data plane and using a random forest algorithm. These approaches encounter major robustness problems when deployed downstream where DDoS traffic converges, and struggle to achieve comprehensive network visibility and effective coordination when deployed upstream.

Multi-Point Detection Methods Involving Router Modifications. Jaqen [27] employs a universal sketch to compute flow statistics and applies a multi-threshold detection approach on programmable switches. Wagner [36] proposes a collaborative architecture that shares information on ongoing amplification DDoS attacks. ChameleMon [38] applies Femat's Little Theorem to sketch for dynamic flow monitoring. Stroboscope [33] enables fine-grained traffic monitoring for ISPs by instructing routers to mirror millisecond-long traffic slices based on high-level monitoring queries and a budget. Newton [43] utilizes dynamic and scalable network-wide query optimization and on-data-plane query management to enable precise, intent-driven traffic monitoring. These methods either require the implementation of new protocols or rely on the deployment of programmable network devices, leading to upgrades that are often impractical due to cost and deployment difficulties.

Deep-Learning-Based Methods. Kitsune [29] utilizes an ensemble of autoencoders to perform online anomaly detection by learning per-packet traffic patterns. DoLLM [25] leverages large language models to detect low-rate, multi-vector DDoS attacks by encoding flow sequences into contextual token embeddings. Trident [42] proposes a modular framework for fine-grained and class-incremental detection of unknown traffic, transforming binary anomaly classification into multiple one-class learning tasks. Kim [23] introduces a frequency-based encoding method for non-numerical features like IP-port combinations, improving anomaly detection capabilities. NetVigil [18] applies graph neural networks and contrastive learning for anomaly detection in data center environments, focusing on extracting security-relevant flow patterns from east-west traffic. While these approaches achieve impressive accuracy, they primarily focus on model precision and overlook the inherent NetFlow reporting delays. Moreover, they are generally too time-consuming to support timely detection in typical NetFlow-based deployments.

9 Conclusion

This paper proposes FlowSentry, a novel server-side framework for accelerating NetFlow-based DDoS attack detection by leveraging cross-router traffic correlation. FlowSentry employs a dual-layer filtering paradigm to reduce the computational overhead and enhance the detection sensitivity. The framework integrates a sliding window mechanism, ADWindow, for efficient data processing and a spatiotemporal analyzer, STAnalyzer, for effective DDoS attack identification using partially reported NetFlow data. Experimental results demonstrate that FlowSentry reduces detection delay by 65.63% while achieving better detection accuracy compared to representative academic and industrial baselines.

10 Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work was supported in part by the National Natural Science Foundation of China under Grant 62132009 and under Grant 62221003. It was also supported in part by Zhongguancun Laboratory.

References

[1] Anup Agarwal, Zaoxing Liu, and Srinivasan Seshan. 2022. HeteroSketch: Coordinating network-wide monitoring in heterogeneous and dynamic networks. In *Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, Renton, WA, 719–741.

[2] Albert Gran Alcoz, Martin Strohmeier, Vincent Lenders, and Laurent Vanbever. 2022. Aggregate-based congestion control for pulse-wave DDoS defense. In *Proceedings of the 2022 ACM SIGCOMM Conference*. ACM, Amsterdam, Netherlands, 693–706.

[3] Paul Almasan, Krzysztof Rusek, Shihan Xiao, et al. 2023. Leveraging Spatial and Temporal Correlations for Network Traffic Compression. arXiv:2301.08962

[4] Manos Antonakakis, Tim April, Michael Bailey, et al. 2017. Understanding the Mirai Botnet. In *Proceedings of the 26th USENIX Security Symposium*. USENIX Association, Vancouver, BC, 1093–1110.

[5] Diogo Barradas, Nuno Santos, Luís Rodrigues, et al. 2021. FlowLens: Enabling Efficient Flow Classification for ML-based Network Security Applications. In *Proceedings of the 28th Network and Distributed System Security Symposium (NDSS)*. Internet Society, San Diego, CA.

[6] Kevin Bock, Abdulrahman Alaraj, Yair Fax, et al. 2021. Weaponizing middleboxes for TCP reflected amplification. In *Proceedings of the 30th USENIX Security Symposium*. USENIX Association, Vancouver, BC, 3345–3361.

[7] Tobias Bühler, Romain Jacob, Ingmar Poese, and Laurent Vanbever. 2023. Enhancing Global Network Monitoring with Magnifier. In *Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, Boston, MA, 1521–1539.

[8] Cisco. [n. d.]. *Cisco IOS NetFlow Configuration Guide*. Cisco Systems. Retrieved in June 2025 from https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html.

[9] Cloudflare. [n. d.]. DDoS threat report for 2024 Q4. Retrieved in June 2025 from https://radar.cloudflare.com/reports/ddos-2024-q4/.

[10] CORERO and Juniper. [n. d.]. 2023 DDoS Threat Intelligence Report. Retrieved in June 2025 from https://www.juniper.net/content/dam/www/assets/analyst-reports/us/en/2023/corero-ddos-threat-intelligence-report.pdf.

[11] Graham Cormode and Shan Muthukrishnan. 2005. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms* 55, 1 (2005), 58–75. doi:10.1016/j.jalgor.2003.12.001

[12] Jisa David and Ciza Thomas. 2019. Efficient DDoS flood attack detection using dynamic thresholding on flow-based network traffic. *Computers & Security* 82 (2019), 284–295. doi:10.1016/j.cose.2019.01.002

[13] Chuanpu Fu, Qi Li, Meng Shen, and Ke Xu. 2021. Realtime Robust Malicious Traffic Detection via Frequency Domain Analysis. In *Proceedings of the 2021 ACM Conference on Computer and Communications Security (CCS)*. ACM, Seoul, South Korea, 3431–3446.

[14] Chuanpu Fu, Qi Li, and Ke Xu. 2023. Detecting unknown encrypted malicious traffic in real time via flow interaction graph analysis. In *Proceedings of the 30th Network and Distributed System Security Symposium (NDSS)*. Internet Society, San Diego, CA.

[15] Harm Griffioen, Kris Oosthoek, Paul van der Knaap, and Christian Doerr. 2021. Scan, test, execute: Adversarial tactics in amplification DDoS attacks. In *Proceedings of the 2021 ACM Conference on Computer and Communications Security (CCS)*. ACM, Seoul, South Korea, 940–954.

[16] Tiago Heinrich, Carlos A. Maziero, Newton C. Will, and Rafael R. Obelheiro. 2022. How DRDoS attacks vary across the globe?. In *Proceedings of the 2022 Internet Measurement Conference (IMC)*. ACM, Nice, France, 760–761.

[17] Rick Hofstede, Václav Bartoš, Anna Sperotto, and Aiko Pras. 2013. Towards real-time intrusion detection for NetFlow and IPFIX. In *Proceedings of the 9th International Conference on Network and Service Management (CNSM)*. IEEE, Zurich, Switzerland, 227–234.

[18] Kevin Hsieh, Mike Wong, Santiago Segarra, Sathiya Kumaran Mani, Trevor Eberl, Anatoliy Panasyuk, Ravi Netravali, Ranveer Chandra, and Srikanth Kandula. 2024. NetVigil: Robust and Low-Cost Anomaly Detection for East-West Data Center Security. In *Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, Santa Clara, CA, 1771–1789.

[19] Huawei. [n. d.]. *NetStream Feature Guide*. Huawei Technologies Co., Ltd. Retrieved in June 2025 from https://support.huawei.com/enterprise/en/doc/EDOC1000178174/986bf11e/overview-of-netstream.

[20] Huawei. 2023. *Global DDoS Attack Status and Trend Analysis in 2023*. Huawei Technologies Co., Ltd. Retrieved in June 2025 from https://e.huawei.com/en/material/networking/security/0c561b8fd2d342999cd402bcecf6d452.

[21] Juniper. [n. d.]. *J-Flow Overview*. Juniper Networks. Retrieved in June 2025 from https://www.juniper.net/documentation/us/en/software/junos/flow-monitoring/topics/concept/flowmonitoring-definitions-solutions.html.

[22] Min Suk Kang, Soo Bum Lee, and Virgil D. Gligor. 2013. The Crossfire Attack. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy (S&P)*. IEEE, San Francisco, CA, 127–141.

[23] Minsong Kim, Woohyuk Jang, JunNyung Hur, and MyungKeun Yoon. 2024. Relative Frequency-Rank Encoding for Unsupervised Network Anomaly Detection. *IEEE/ACM Transactions on Networking* 32, 4 (2024), 3453–3467. doi:10.1109/TNET.2024.3391396

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

[25] Qingyang Li, Yihang Zhang, Zhidong Jia, Yannan Hu, Lei Zhang, Jianrong Zhang, Yongming Xu, Yong Cui, Zongming Guo, and Xinggong Zhang. 2024. DoLLM: How Large Language Models Understanding Network Flow Data to Detect Carpet Bombing DDoS. arXiv:2404.15766 [cs.NI]

[26] Yuliang Li, Rui Miao, Changhoon Kim, and Minlan Yu. 2016. FlowRadar: A Better NetFlow for Data Centers. In *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, Santa Clara, CA, 311–324.

[27] Zaoxing Liu, Hun Namkung, Georgios Nikolaidis, Jeongkeun Lee, Changhoon Kim, Xin Jin, Vladimir Braverman, Minlan Yu, and Vyas Sekar. 2021. Jaqen: A High-Performance Switch-Native Approach for Detecting and Mitigating Volumetric DDoS Attacks with Programmable Switches. In *Proceedings of the 30th USENIX Security Symposium*. USENIX Association, Vancouver, BC, 3829–3846.

[28] MaxMind, Inc. [n. d.]. *GeoIP Dataset Documentation*. MaxMind, Inc. Retrieved in 2025 from https://dev.maxmind.com/geoip/.

[29] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. 2018. Kitsune: an ensemble of autoencoders for online network intrusion detection. In *Proceedings of the 2018 Network and Distributed System Security Symposium (NDSS)*. Internet Society, San Diego, CA.

[30] Yuqi Qing, Qilei Yin, Xinhao Deng, Yihao Chen, Zhuotao Liu, Kun Sun, Ke Xu, Jia Zhang, and Qi Li. 2024. Low-quality Training Data Only? A Robust Framework for Detecting Encrypted Malicious Network Traffic. In *Proceedings of the 31st Network and Distributed System Security Symposium (NDSS)*. Internet Society, San Diego, CA.

[31] Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak, and Ali Ghorbani. 2019. Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy. In *Proceedings of the 2019 IEEE International Carnahan Conference on Security Technology (ICCST)*. IEEE, Chennai, India, 1–8.

[32] Gagan Somashekar, Karan Tandon, Anush Kini, Prithvi Ravi, Shubham Choudhary, Chiranjeeb Buragohain, and Aditya Akella. 2024. OPPerTune: Post-Deployment Configuration Tuning of Services Made Easy. In *Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, Santa Clara, CA, 1101–1120.

[33] Olivier Tilmans, Tobias Bühler, Ingmar Poese, and Laurent Vanbever. 2018. Stroboscope: Declarative Network Monitoring on a Budget. In *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, Renton, WA, 467–482.

[34] Martino Trevisan, Danilo Giordano, Idilio Drago, Marco Mellia, and Maurizio Munafo. 2018. Five Years at the Edge: Watching Internet from the ISP Network. In *Proceedings of the 14th International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*. ACM, Heraklion, Greece, 1–12.

[35] Muhammad Fahad Umer, Muhammad Sher, and Yaxin Bi. 2017. Flow-based intrusion detection: Techniques and challenges. *Computers & Security* 70 (2017), 238–254. doi:10.1016/j.cose.2017.05.009

[36] Daniel Wagner, Daniel Kopp, Matthias Wichtlhuber, et al. 2021. United we stand: Collaborative detection and mitigation of amplification DDoS attacks at scale. In *Proceedings of the 2021 ACM Conference on Computer and Communications Security (CCS)*. ACM, Seoul, South Korea, 970–987.

[37] WIND. [n. d.]. MAWI Working Group Traffic Archive. Retrieved in June 2025 from http://mawi.wide.ad.jp/mawi/.

[38] Kaicheng Yang, Yuhan Wu, Ruijie Miao, and Minlan Yu. 2023. ChameleMon: Shifting Measurement Attention as Network State Changes. In *Proceedings of the ACM SIGCOMM 2023 Conference*. ACM, New York, NY, 881–903.

[39] Tong Yang, Jie Jiang, Peng Liu, et al. 2018. Elastic sketch: Adaptive and fast network-wide measurements. In *Proceedings of the 2018 ACM SIGCOMM Conference*. ACM, Budapest, Hungary, 561–575.

[40] Tong Yang, Haowei Zhang, Jinyang Li, and Minlan Yu. 2019. HeavyKeeper: An Accurate Algorithm for Finding Top-k Elephant Flows. *IEEE/ACM Transactions on Networking* 27, 5 (2019), 1845–1858. doi:10.1109/TNET.2019.2933868

[41] ZDNet. [n. d.]. GitHub hit with the largest DDoS attack ever seen. Retrieved in June 2025 from https://www.zdnet.com/article/github-was-hit-with-the-largest-ddos-attack-ever-seen/.

[42] Ziming Zhao, Zhaoxuan Li, Zhuoxue Song, Wenhao Li, and Fan Zhang. 2024. Trident: A Universal Framework for Fine-Grained and Class-Incremental Unknown Traffic Detection. In *Proceedings of the ACM Web Conference (WWW)*. ACM, Singapore, 1608–1619.

[43] Yu Zhou, Dai Zhang, Kai Gao, Xiangyu Gao, Hongxin Hu, and Mingwei Xu. 2020. Newton: Intent-driven Network Traffic Monitoring. In *Proceedings of the 16th International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*. ACM, Barcelona, Spain, 295–308.