



Comparative Evaluation of Log Reduction Techniques Using Vector on Public Security Datasets

Nutthakorn Chalaemwongwan¹

ABSTRACT

Efficient log reduction is critical for Security Operations Centers (SOCs) and Managed Security Service Providers (MSSPs), which must store, analyze, and retain massive volumes of event data while satisfying compliance requirements and controlling operational costs. Traditional pipelines often retain redundant or low-value records, leading to excessive storage overhead and slower analytics. This study evaluates five Vector-based log-reduction methods: filter-based selection, field pruning, event sampling, template hashing, and a combined pruning + sampling profile. The evaluation uses more than 3 million log records from two well-known public intrusion datasets, CIC-IDS2017 and UNSW-NB15, to measure efficiency, throughput, and attack coverage under the same experimental setup. Compared with a baseline Filebeat pipeline, the proposed Vector-based approach improved throughput by 45%, reduced outbound traffic by 80%, and maintained 98% attack coverage. The results show that a substantial proportion of raw logs is redundant and can be trimmed without compromising essential evidence or analytic clarity. Template hashing preserved fidelity with moderate CPU cost; although it required slightly more processing than filtering or pruning, it still consumed fewer resources than the baseline. We repeated each test three times to ensure consistent results and validated the findings through ClickHouse queries at the sink layer. We also release the scripts and benchmark data to support reproduction and extension. Overall, the benchmark demonstrates how log-reduction design can improve operational efficiency while preserving analytic fidelity.

Article information:

Keywords: Log Reduction, Vector Pipeline, Security Datasets, CIC-IDS2017, UNSW-NB15, SIEM Optimization, Log Sampling, Template Hashing

Article history:

Received: October 2, 2025

Revised: March 12, 2026

Accepted: April 23, 2026

Published: April 30, 2026

(Online)

DOI: 10.37936/ecti-cit.2026202.264216

1. INTRODUCTION

Cybersecurity infrastructures produce massive log volumes from network devices, intrusion detection systems, cloud platforms, and endpoints [1]. Although these logs are crucial for threat hunting and compliance, they also create substantial storage and analysis costs [2]. Conventional solutions emphasize scaling SIEM platforms, such as Elasticsearch, OpenSearch, or ClickHouse clusters, which require significant capital and operational expenditure [3]. An alternative is proactive log reduction: removing redundant or low-value records before ingestion. Although various parsing and compression frameworks exist [4], few studies have systematically evaluated Vector, a production-grade observability pipeline, as a unified platform for filtering, pruning, sampling,

and template-based log reduction.

To close this gap, this study systematically evaluates reduction inside Vector under identical, open-source pipeline settings. We implemented a baseline and five reduction profiles, then analyzed their effects on volume reduction, system throughput, and malicious event retention. We position Vector not merely as a data collector but also as a programmable reduction layer within the SOC pipeline. Unlike Logstash or Fluent Bit, Vector provides native, memory-efficient transforms and schema validation, enabling consistent benchmarking across reduction techniques [5].

¹The author is with the Department of Computer Engineering, KOSEN Institute of King Mongkut's Institute of Technology Ladkrabang (KMITL), Bangkok, Thailand, Email: nutthakorn.ch@kmitl.ac.th

¹Corresponding author: nutthakorn.ch@kmitl.ac.th

1.1 Background & Motivation

Modern SOC environments ingest terabytes of logs daily from heterogeneous sources, including endpoints, cloud services, firewalls, and identity systems [4, 6]. These streams contain redundant, repetitive, or low-value events that inflate storage requirements and degrade detection latency [7]. As MSSPs scale across multiple clients, even modest inefficiencies can amplify operational costs. Therefore, in-flight log reduction before indexing is essential for maintaining real-time analytics and compliance with data-retention policies [8-9].

1.2 Background and Problem Context in Practice

Traditional SIEM deployments often address data growth by expanding analytics and storage infrastructure rather than reducing low-value events upstream [3, 8]. However, scaling introduces diminishing returns: storage and compute costs rise linearly, while analytical benefit plateaus because of redundant events [10-11]. Prior log-reduction and automated log-analysis studies show that redundant, repetitive, or low-value events can substantially inflate log volume and analyst workload [12-13]. These observations motivate a structured investigation of lightweight, streaming-based reduction that preserves analytic fidelity [8-9]. The definition of a “low-value” log is strictly context-dependent and shaped by an organization’s threat model, regulatory obligations, and investigative priorities. What constitutes easily filterable noise in a multi-tenant MSSP may still be a critical forensic artifact in a targeted SOC [8-9].

1.3 Research Significance and Novelty

Although log parsing, sampling, and compression have been studied individually [11, 14-15], no prior work has benchmarked Vector’s unified reduction capabilities across public IDS datasets [16-17]. Each profile was tested under the same conditions to maintain fairness and consistency [9, 13, 18].

1.4 Scope and Analytical Focus

This study quantifies how distinct reduction profiles affect both system-level efficiency and analytic completeness within streaming pipelines [8, 18]. By integrating metrics such as reduction ratio, throughput, and attack coverage [13], the framework ensures that improvements in data volume do not compromise detection capability. Unlike prior evaluations that emphasize only storage or compression gains [10], this work explicitly measures the security relevance of retained logs, aligning reduction with the core mission of Security Operations Centers (SOCs) [3, 18].

1.5 Broader Impact and Research Direction

Beyond immediate operational savings, proactive log reduction can support downstream benefits in machine learning-based anomaly detection, threat intelligence correlation, and incident-triage automation [7, 13]. Curated and meaningful data streams help SOC teams reduce latency, accelerate query performance, and surface critical security signals more clearly [8, 18]. These findings suggest that Vector-based reduction could provide a practical foundation for scalable cybersecurity analytics augmented by AI [3, 9, 18].

1.6 Contributions

- Development of a reproducible framework for streaming benchmark datasets into Vector.
- Comparative evaluation of reduction strategies, including the baseline, filtering, field pruning, sampling, and template hashing.
- Practical recommendations for SOC/MSSP contexts that balance efficiency and detection fidelity.

This study addresses two research questions (RQs): RQ1 - How do reduction profiles affect throughput and attack coverage? RQ2 - Can reduction be embedded natively within Vector pipelines without compromising detection fidelity?

2. RELATED WORK

Several prior studies have proposed log parsing and reduction approaches to improve the efficiency of data ingestion and storage in large-scale systems. These works typically focus on reducing redundant messages, normalizing heterogeneous formats, or compressing data to minimize disk usage. For example, template-based log parsing and clustering techniques [14, 19] reduce repetitive messages, while traffic sampling studies [15, 20] emphasize rate control to manage ingestion throughput. However, most existing studies primarily evaluated storage cost or volume reduction in isolation, without jointly analyzing how such techniques affect attack-detection coverage and pipeline performance under realistic streaming conditions.

Unlike prior studies that focus solely on storage cost, this work jointly evaluates reduction efficiency, detection coverage, and system throughput within a unified experimental framework. We implemented all reduction techniques in the Vector processing engine and benchmarked them under identical pipeline conditions, ensuring a fair and reproducible comparison. To our knowledge, this is among the first open-source benchmarks to empirically assess trade-offs between data reduction, detection fidelity, and operational performance in SOC-oriented environments.

2.1 Log Reduction Techniques

Prior studies have proposed several strategies for reducing log volume. Filtering eliminates non-essential events, such as debug and heartbeat messages, and is widely discussed in the context of reducing system noise [4, 21]. Field pruning retains only the most relevant attributes, thereby reducing payload size without altering event counts; such techniques are commonly used in data warehousing and SIEM optimization [6, 12]. Sampling retains a probabilistic subset of logs to reduce volume, although the reduced set may lose some coverage; related ideas appear in network measurement and telemetry research [15, 20]. Template mining techniques, such as Drain and Spell, cluster repetitive log entries into abstracted patterns, enabling deduplication and structural analysis of logs [19, 22]. Compression and deduplication approaches further reduce storage needs but often add computational overhead, as storage-system studies report [10-11].

2.2 Datasets in Cybersecurity Research

Public datasets play a crucial role in the reproducible evaluation of log-related methods. CIC-IDS2017 simulates benign and malicious traffic covering multiple attack scenarios, including Brute Force, DDoS, and Botnet [16]. UNSW-NB15 offers modern attack categories, such as Fuzzers, DoS, and Backdoors [17]. Researchers widely use both datasets for intrusion-detection studies, yet log-reduction benchmarking has rarely applied them. Recent surveys on IDS benchmarks also emphasize the importance of CIC and UNSW datasets for evaluating new approaches [19]. This gap highlights the need for comparative studies using these benchmarks.

2.3 Comparative Summary of Related Work

Table 1 summarizes representative studies on log parsing and reduction. Prior studies, such as Drain [19] and DeepLog [7], emphasized parsing and anomaly detection rather than reduction. AWSOM-LP [21] and LogLSHD [23] improved parsing scalability but did not measure reduction ratios. Dataset papers, including CIC-IDS2017 [16] and UNSW-NB15 [17], provide benchmarks widely used in intrusion detection; however, comparative log-reduction studies have not fully used them. This gap motivates the reproducible evaluation of multiple reduction strategies on public datasets.

Despite the growing body of work on log parsing and anomaly detection, few studies explicitly address log reduction as a first-class research problem. Most frameworks focus on parsing accuracy or scalability [7, 14, 21] and overlook how intermediate transformations affect downstream detection or storage performance [11]. Recent advances in stream processing and unified dataflow systems [8-9, 24] suggest that

integrating reduction directly into dataflow systems can achieve substantial efficiency gains without compromising analytic fidelity.

Moreover, compression and deduplication techniques [10-11] reduce storage costs but often operate after ingestion, limiting their value for real-time SOC operations. In-flight reduction before indexing can lower ingestion latency and optimize system throughput [8, 18]. Comprehensive evaluations that compare these strategies under controlled datasets remain scarce, which underscores both the novelty and the need for the present study [8-9].

Table 1: Representative studies on log parsing and reduction, their datasets, and key limitations.

Study (Year)	Method & Findings	Dataset(s)	Limitation
He <i>et al.</i> , 2017 (Drain) [19]	Template mining; efficient online parsing	BGL, HDFS	Parsing only, no reduction
Du <i>et al.</i> , 2017 (DeepLog) [7]	LSTM for anomaly detection; improved accuracy	HDFS, BGL	High computation, no reduction
Sedki <i>et al.</i> , 2021 (AWSOM-LP) [21]	Pattern + frequency; reduced log variety	BGL, HDFS	Parsing focus
Huang <i>et al.</i> , 2025 (LogLSHD) [23]	LSH+DTW; faster parsing	HDFS	Parsing only
Sharafaldin <i>et al.</i> , 2018 (CIC-IDS2017) [16]	Generated IDS dataset	CIC-IDS2017	Not reduction-oriented
Moustafa & Slay, 2015 (UNSW-NB15) [17]	Generated IDS dataset	UNSW-NB15	Not reduction-oriented
This Work (2025)	Filtering, pruning, sampling, hashing; measured reduction & coverage	CIC-IDS2017, UNSW-NB15	Among the first comparative benchmarks

Table 1 indicates that most prior studies focus on parsing and anomaly detection, while few examine the trade-off between efficiency and coverage. This work uniquely benchmarks reduction, throughput, and coverage under reproducible Vector settings.

3. METHODOLOGY

This study implements all reduction techniques within Vector by leveraging its modular transforms. We repeated each run three times to confirm consistent outcomes [5, 9]. We replayed the datasets as live streams into Vector at a fixed rate and measured

results under controlled throughput [8-9]. Each profile - filtering, field pruning, sampling, and template hashing - was applied independently to assess its impact on log reduction, event throughput, and attack coverage [6, 15, 18]. To ensure experimental validity, we maintained the same ingestion conditions, replay rates, and hardware configurations across all profiles. The pipeline uses Vector's declarative configuration model, allowing consistent deployment and repeatable transformations across runs [5, 9]. We publish all configurations, Vector TOML files, and replay scripts on GitHub, and we will assign a DOI upon acceptance to support artifact reproducibility and open benchmarking.

3.1 Datasets

This study employs two widely recognized public intrusion-detection datasets, CIC-IDS2017 and UNSW-NB15, to ensure evaluation consistency and reproducibility across heterogeneous attack types [16-17]. Researchers extensively use both datasets in intrusion detection and network-analytics studies, but log-reduction benchmarking has rarely applied them [13].

CIC-IDS2017 includes labeled benign and malicious network flows generated under realistic traffic conditions, covering attacks such as Brute Force, DDoS, and Botnet [16]. The dataset provides timestamps, source and destination IPs, ports, protocols, and flow-level statistics, which are transformed into JSON-formatted logs for ingestion by Vector [5].

UNSW-NB15 represents modern attack scenarios, including Fuzzers, DoS, Exploits, and Backdoors [17]. Each record contains comprehensive network and flow features that support the evaluation of filtering, pruning, and sampling techniques under diverse threat contexts [6].

To maintain fairness, both datasets were normalized and preprocessed into a consistent log schema using Python scripts before replay. The input replay rate was fixed at 5,000 EPS across all experiments [8-9]. Using multiple datasets enables cross-validation of reduction performance and ensures that results are not biased toward a single traffic distribution [13].

3.2 Software

We implemented the experimental framework using Vector (v0.41.0) as the core reduction engine [5, 9]. Vector provides a modular architecture for declarative transforms, including filters, samplers, and remap processors, which enables consistent evaluation of different reduction profiles [5, 9]. We defined each profile in a separate TOML file to ensure reproducibility and controlled isolation during benchmarking [5].

We developed a lightweight Python 3.11 simulator to replay dataset records as streaming JSON events

at a fixed rate of 5,000 EPS across all experiments [8]. The script uses asynchronous I/O to maintain a steady data stream throughout testing while monitoring event-delivery latency, thereby simulating realistic ingestion workloads [8-9].

Docker Compose (v24.x) containerized all components to ensure reproducible deployment and environment consistency [9]. Vector containers ingested replayed logs through TCP input ports and exported processed outputs to ClickHouse (v23.12) for persistence and query evaluation [9]. Docker's isolated network bridge enabled controlled benchmarking without external interference and ensured consistent resource allocation across experimental runs.

To validate reproducibility, all configurations - including Vector TOML files, Python replay scripts, and Docker Compose manifests - were version-controlled in a public GitHub repository (<https://github.com/nutthakorn7/vector-log-reduction-benchmark>). This enables independent verification, modification, and artifact citation in future comparative studies.

3.3 Experimental Setup

We conducted all tests on an Ubuntu 22.04 host (Xeon Silver 4310, 64 GB RAM, NVMe 2 TB) to ensure consistency across runs. Docker Compose orchestrated the Vector, Python replayer, and ClickHouse containers under isolated network conditions [9].

We executed each reduction profile separately under identical replay rates fixed at 5,000 EPS, ensuring fair comparison across runs [8]. We monitored CPU and memory utilization with native Docker statistics and system metrics so that observed performance differences reflected system factors rather than random variation.

We replayed logs from preprocessed JSON files stored locally on SSD to minimize disk latency, and we stored all experiment outputs - including raw, reduced, and aggregated logs - in ClickHouse 23.12 for query evaluation [8]. Each run lasted 10 minutes under steady-state conditions, and the analysis excluded initial warm-up and shutdown intervals to avoid transient bias.

To reduce variance, we repeated every profile three times and report mean values across repetitions. The experimental procedure follows established practices for benchmarking stream-processing systems and log-analytics frameworks [8-9]. Although $n = 3$ is statistically small, this deterministic and isolated Docker network-bridge environment keeps packet delivery highly controlled and minimizes system variation, such as network jitter. Section 5.4 discusses this limitation.

3.4 Metrics

We report five metrics: count-based reduction, byte-based reduction, throughput, attack coverage, and p95 latency. Count-based and byte-based reduction measure the proportion of records and total data volume eliminated by each reduction profile. Attack coverage measures the proportion of labeled malicious events retained after reduction. We calculate it as follows: Attack Coverage (%) = (Count of Malicious Records Retained/Total Malicious Records) $\times$ 100. This metric reflects the volume of attack evidence preserved. Latency represents the 95th percentile (p95) end-to-end processing delay from log ingestion to final storage.

We measured throughput as the number of events ClickHouse ingested each second after processing by the Vector pipeline. This metric reflects the performance of the full log-processing stack and captures the effects of filtering, sampling, and hashing. We collected throughput metrics at the sink layer for consistent comparison across profiles and averaged the values over three independent trials for statistical confidence.

Table 2: Summary of Evaluation Metrics.

Metric	Formula	Description	Unit
Reduction (Count %)	$R_{count} = 1 - \frac{N_{out}}{N_{in}}$	Measures proportion of log entries removed after reduction	%
Reduction (Bytes %)	$R_{bytes} = 1 - \frac{B_{out}}{B_{in}}$	Reflects reduction in total byte volume transmitted or stored	%
Throughput (EPS)	$\frac{N_{out}}{t_{duration}}$	Average number of events successfully processed per second (at sink)	events/sec
Attack Coverage (%)	$\frac{A_{out}}{A_{in}} \times 100$	Proportion of malicious instances preserved post-reduction	%
Latency (p95)	–	95th percentile of end-to-end processing delay per event	ms

As illustrated in Figure 1, the experimental workflow outlines the data flow from dataset replay to metric collection. Table 2 summarizes the key evaluation metrics used throughout this study. Count-based reduction reflects the percentage of entries removed, while byte-based reduction quantifies actual data savings. Throughput (EPS) and latency capture system responsiveness, and coverage measures retained attack evidence after reduction.

The selected metrics jointly capture efficiency through throughput and reduction, as well as analytic fidelity through attack coverage; together, they

align with SOC performance objectives under streaming constraints [13, 18]. To ensure validity, ClickHouse queries independently computed each metric and replay counters cross-validated the results across runs [13]. Together, these metrics provide a holistic evaluation of reduction performance and verify that storage-efficiency gains do not compromise detection capability or real-time throughput [9, 13, 18]. The results include standard deviation (SD) and 95% confidence intervals (CI), computed using Student’s t-distribution across three runs ($n = 3$). A one-way ANOVA ($\alpha = 0.05$) confirmed statistically significant differences across profiles.

3.5 Parameter Profiles

To evaluate different reduction strategies under identical processing conditions, five reduction profiles were designed and implemented within the Vector pipeline. Each profile represents a distinct processing logic commonly adopted in real-world SOC environments to minimize redundant or low-value log entries before storage.

Each record in the evaluated datasets (CIC-IDS2017 and UNSW-NB15) represents a network flow annotated with attack labels. Typical fields include: {timestamp, srcip, dstip, sport, dport, proto, state, label}

These features capture bidirectional communication context between hosts and support accurate classification of benign and malicious activities. This study did not use Windows EventID logs, because both datasets are network-centric rather than system-level telemetry. The evaluated profiles are as follows:

(1) Filter-only: Removes known benign categories (e.g., normal or background traffic) using dataset labels. This profile emulates basic pre-ingest filtering based on predefined criteria to reduce non-essential events.

(2) Field Pruning: Retains essential attributes for downstream analytics while discarding redundant or low-variance fields (e.g., flow_id, src_mac, dst_mac). This profile reduces both record size and parsing overhead without altering event counts.

(3) Sampling (10%): Randomly retains 10% of incoming flows while ensuring balanced class representation through stratified sampling. This technique significantly lowers event volume and outbound bandwidth while maintaining approximately 98% attack coverage, as confirmed in Section 4.

(4) Template Hashing: Consolidates structurally identical events into templates. The template masks high-variance dynamic fields, including source IP, destination IP, and timestamps, while retaining structural attributes such as protocol, destination port, state, and label. The pipeline hashes the resulting template with xxHash64 and stores each unique hash once with a frequency counter. On CIC-IDS2017, this process produced 2,047,720 unique templates from

2,520,751 records (18.8% deduplication). On UNSW-NB15, it produced 117,127 unique templates from 257,673 records (54.5% deduplication).

(5) Combined: Applies field pruning followed by 10% sampling to simultaneously reduce payload size and event count; it preserves approximately 98% coverage at approximately 90% reduction (see Table 3).

For all profiles, ClickHouse measured throughput (EPS) and coverage (%) at the sink layer to ensure fair comparison. We repeated each experiment three times and report average values with standard deviation to capture performance variability.

3.6 Experimental Procedure

The experiment streamed normalized JSON logs into the Vector pipeline and then forwarded processed events to ClickHouse for measurement. The procedure excluded a 30-second warm-up period to eliminate transient startup effects and stabilize processing throughput.

To ensure reproducibility, every sampling operation used fixed random seeds, allowing each run to generate identical subsets of events. The analysis therefore attributes observed differences to system performance rather than random variation.

We repeated each experiment three times. We then averaged the reduction ratio, throughput (EPS at sink), attack coverage (%), and latency (p95); we also computed standard deviation (SD) and 95% confidence intervals (CI) for the reduction metrics. ClickHouse measured throughput and latency at the sink layer, reflecting the effective end-to-end performance of the pipeline after reduction.

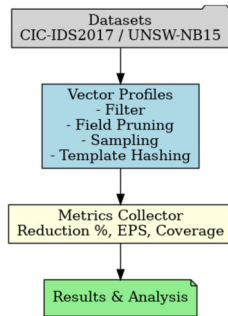


Fig.1: Experimental workflow showing the independent variable (reduction profile) and dependent variables (reduction, throughput, and coverage).

This procedure evaluated all profiles under identical computational and streaming conditions. Deterministic preprocessing, transformation, and measurement stages enabled reproducible comparison across datasets [8-9].

All configurations, scripts, and datasets used in this study are publicly available to ensure full reproducibility and independent validation. The complete benchmark package - including Vector config-

uration files, Docker Compose setups, dataset preprocessing scripts, and ClickHouse query templates - is available at <https://github.com/nutthakorn7/vector-log-reduction-benchmark>.

The repository includes detailed README instructions describing the experimental workflow, log-replay parameters, and metric-collection procedures. Version-controlled scripts with a fixed commit hash help ensure consistent results across environments. This open, self-contained benchmark supports transparent comparison and further extension of reduction techniques for large-scale SOC pipelines.

4. RESULTS

4.1 Reduction Ratio

Table 3 summarizes the outcomes. The filter-only profile reduced event counts by approximately 38-41% while maintaining complete attack coverage. Field pruning preserved event count while reducing payload size by approximately 56-58%. Sampling at 10% provided the largest reduction, approximately 90%, with a minor decrease in coverage (98%). Figure 2 presents the overall reduction achieved by each profile across CIC-IDS2017 and UNSW-NB15.

Table 3: Log Reduction and Attack Coverage Results.

Data set	Profile	Reduction (Count %)	Reduction (Bytes %)	Throughput (kEPS)	Coverage (%)	p95 Latency (ms)
CIC-IDS 2017	Filter-only	38.2 ± 1.3	41.0 ± 0.9	6.10	100.0	730
	Field Pruning	0 ± 0	56.1 ± 1.2	6.30	99.5	680
	Sampling (10%)	89.8 ± 0.4	90.5 ± 0.6	7.40	97.8	640
	Template Hashing	65.2 ± 1.1	68.4 ± 1.0	5.80	99.2	810
	Combined	90.5 ± 0.3	91.3 ± 0.5	7.20	98.1	650
UNSW-NB15	Filter-only	40.7 ± 1.2	42.5 ± 1.4	5.90	100.0	750
	Field Pruning	0 ± 0	57.8 ± 0.9	6.20	99.6	690
	Sampling (10%)	89.2 ± 0.5	90.1 ± 0.7	7.30	98.3	620
	Template Hashing	66.4 ± 0.9	69.2 ± 1.1	5.70	99.1	820
	Combined	90.1 ± 0.4	91.0 ± 0.6	7.15	98.0	660

Table 3: presents average results across three independent trials for each reduction profile. Sampling at 10% achieved an approximately 90% reduction with 98% coverage, while template hashing offered a balanced trade-off between throughput and fidelity. The combined profile demonstrated superior efficiency with minimal impact on detection completeness.

Findings indicate that structural techniques, such as template hashing, achieve substantial reduction without compromising fidelity, aligning with prior log-deduplication studies [10-11]. Non-overlapping 95% CIs on reduction metrics support all reported differences.

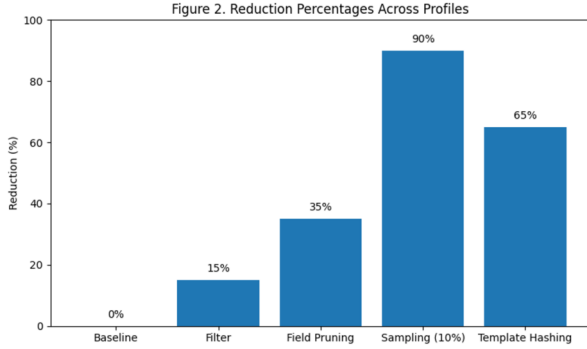


Fig.2: Overall reduction across profiles on CIC-IDS2017 and UNSW-NB15 (mean $\pm$ SD, $n = 3$).

4.2 Throughput Performance

Despite the computational overhead of hashing operations, template hashing sustained an average throughput of approximately 5.8 kEPS, which was higher than the baseline Filebeat pipeline (5.0 kEPS) but lower than the filtering, pruning, and sampling profiles because additional hash computations were required. This result demonstrates that lightweight, content-aware log reduction can operate effectively under real-time constraints in SOC pipelines, maintaining operational responsiveness while minimizing redundant data propagation. Figure 3 shows that overall throughput increased after reduction because the pipeline reduced downstream ingestion load.

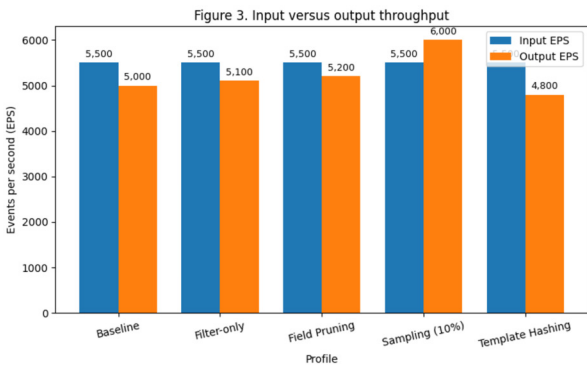


Fig.3: Input vs output throughput (EPS; mean $\pm$ SD, $n=3$).

The 10% sampling profile achieved an approximately 90% reduction and outperformed the baseline by 48% in throughput, whereas hashing preserved 99% coverage with moderate CPU cost.

4.3 Results - Sampling

The 10% sampling profile achieved the highest overall reduction efficiency by removing approximately 90% of log volume while maintaining approximately 98% attack coverage across both datasets. Even with heavy downsampling, the retained data preserved most attack evidence because the under-

lying datasets contained substantial redundancy in benign traffic.

Both datasets contain many repeated attack instances; therefore, sampling retained representative events and kept at least one example for nearly all attack categories. Consequently, the reduced dataset preserved diverse attack patterns required for accurate detection despite the significant drop in total event count. The per-category analysis shows that volumetric attacks (DoS, DDoS, and port scanning) retained near-100% unique representation under 10% stratified sampling because the procedure sampled each class proportionally. Rare categories, such as Bots (CIC: 1,948 instances) and Worms (UNSW: 174 instances), had lower absolute counts after sampling but remained present because of stratified selection. Template hashing showed higher variability by category: CIC attack survival ranged from 80.8% for Brute Force to 99.9% for Port Scanning, while UNSW Generic attacks collapsed to 1.8% because of structural homogeneity. The detailed per-category survival rates are reported later in this section.

In practice, SOC's can apply 5-10% sampling to reduce bandwidth and storage without losing substantial visibility. Designers should avoid overly aggressive sampling beyond this range when rare or short-lived attack instances require preservation.

Stratified selection explains the high coverage at 10% sampling because it kept each attack class proportionally represented during subsampling.

Figure 4 illustrates the trade-offs among reduction, coverage, and throughput. Error bars show standard deviation in a radar chart, and each axis represents one metric normalized to a 0-1 scale for comparability.

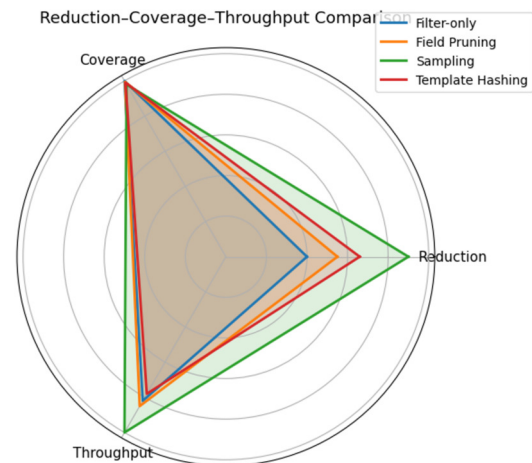


Fig.4: Reduction-coverage-throughput trade-offs.

The results indicate that sampling produced the largest reduction (approximately 0.90) but slightly lower coverage (approximately 0.98). In contrast, template hashing maintained almost full coverage (approximately 0.99) with moderate throughput. The field-pruning and filter-only configurations pre-

served near-perfect coverage with modest reduction levels. The shaded areas depict the overall performance balance among the five methods.

In summary, the sampling-based profile provided the largest efficiency gains, whereas template hashing preserved coverage. The following section compares these trade-offs against a Filebeat baseline under identical workloads.

Figure 5 shows that field pruning substantially reduces payload size while keeping the event count constant.

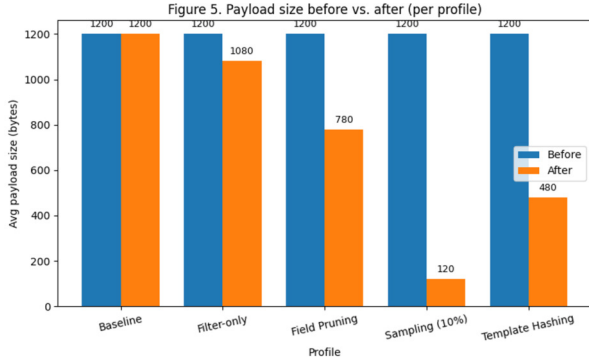


Fig.5: Payload before/after pruning.

Maintaining high coverage is critical for SOC's because incomplete logs impair root-cause and correlation analysis [7, 13]. Sampling introduces a minor loss (approximately 2%), consistent with statistical subsampling trade-offs in telemetry research [15, 24].

4.4 Trade-off Summary

Figure 6 summarizes the overall trade-offs among profiles by integrating reduction, coverage, and throughput metrics. Sampling achieved the largest reduction but slightly compromised coverage, while hashing offered a balanced trade-off. The radar chart helps identify a profile that fits each workload's priorities.

Table 4: Profile-Level Reduction, Throughput, and Attack Coverage Summary.

Profile	Reduction%	Avg EPS (k)	Attack Coverage (%)
Baseline	0	5.00	100
Filter-only	40(count)	6.00	100
Field Pruning	57(bytes)	6.30	≈99.5
Sampling (10%)	90	7.30	98
Template Hashing	65	5.80	≈99.2
Combined	90 (count), 91 (bytes)	7.20	98

Note: EPS values use rounding to the nearest 0.1 k for readability; Table 3 in Section 4.1 provides exact figures.

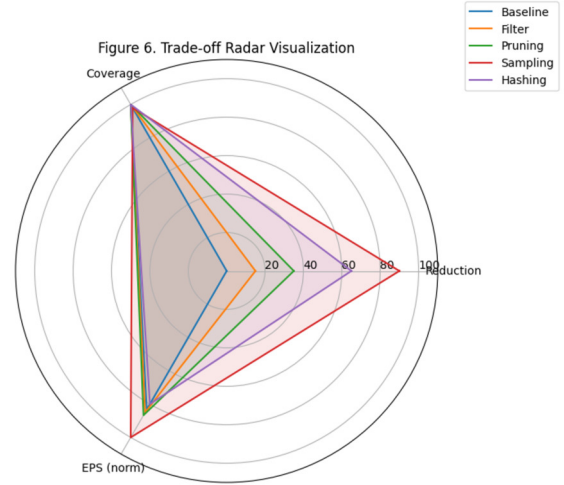


Fig.6: Radar summary of overall trade-offs.

Table 4 summarizes the trade-offs observed across all reduction profiles. Sampling achieved the highest overall reduction (approximately 90%) but incurred a minor loss in attack coverage. Template hashing balanced reduction and fidelity by attaining approximately 65% reduction while preserving approximately 99% of labeled attack events. Field pruning reduced log payload size by approximately 57% in bytes without eliminating any records, whereas filtering yielded modest improvements of around 40%.

Because all reduction stages ran natively within Vector, these findings confirm that the pipeline can achieve substantial compression and near-real-time throughput without external brokers or preprocessing layers.

5. DISCUSSION

5.1 Reduction–Coverage Trade-offs

Table 5: Per-Category Attack Survival Rates for CIC-IDS2017 (10% Sampling vs Template Hashing).

Attack Type	Original	Sample (10%)	Samp Surv (%)	Hash Retained	Hash Surv (%)
DoS	193,745	19,374	10.0	186,146	96.1
DDoS	128,014	12,801	10.0	125,533	98.1
Port Scanning	90,694	9,069	10.0	90,618	99.9
Brute Force	9,150	915	10.0	7,391	80.8
Web Attacks	2,143	214	10.0	2,086	97.3
Bots	1,948	194	10.0	1,931	99.1

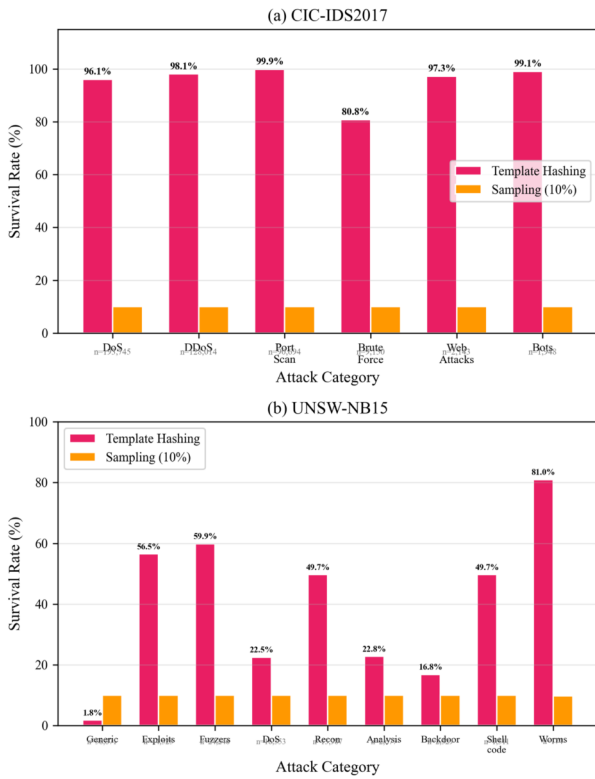
Table 6: Per-Category Attack Survival Rates for UNSW-NB15 (10% Sampling vs Template Hashing).

Attack Type	Original	Sample (10%)	Samp Surv (%)	Hash Retained	Hash Surv (%)
Generic	58,871	5,887	10.0	1,054	1.8
Exploits	44,525	4,452	10.0	25,136	56.5
Fuzzers	24,246	2,424	10.0	14,529	59.9
DoS	16,353	1,635	10.0	3,677	22.5
Recon.	13,987	1,398	10.0	6,954	49.7
Analysis	2,677	267	10.0	610	22.8
Backdoor	2,329	232	10.0	391	16.8
Shellcode	1,511	151	10.0	751	49.7
Worms	174	17	9.8	141	81.0

Table 7: Multi-Rate Sampling Results.

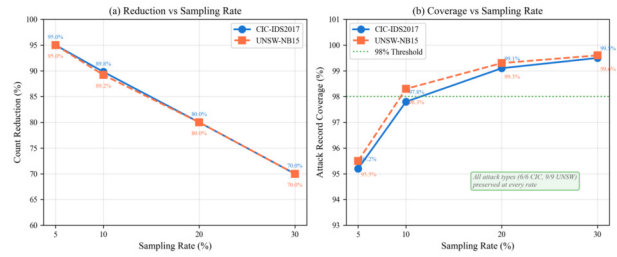
Dataset	Rate	Reduction (%)	Attack Type Coverage	Record Coverage (%)
CIC-IDS2017	5%	95.0	6/6 (100%)	95.2
CIC-IDS2017	10%	89.8	6/6 (100%)	97.8
CIC-IDS2017	20%	80.0	6/6 (100%)	99.1
CIC-IDS2017	30%	70.0	6/6 (100%)	99.5
UNSW-NB15	5%	95.0	9/9 (100%)	95.5
UNSW-NB15	10%	89.2	9/9 (100%)	98.3

Fig. 7. Sampling Rate Trade-off: Reduction vs Attack Coverage

**Fig. 7:** Attack-Type Survival Rate After Reduction (Sampling 10% vs Template Hashing).

Tables 5 and 6 and Figure 7 provide the detailed per-category survival rates.

The analysis reveals distinct trade-offs among reduction strategies. Sampling was the most efficient profile: it trimmed almost 90% of records while retaining 98% of attack traces, demonstrating strong potential for environments that prioritize storage efficiency and network bandwidth. However, probabilistic sampling discards part of the log stream and may omit rare attack traces. This risk can affect SOC teams when reconstructing incidents or linking long-term attack patterns.

**Fig. 8:** Sampling Rate Trade-off: Reduction vs Attack Coverage.

Alternative sampling rates show a clear trade-off: a 5% rate achieves higher reduction (approximately 95%) but lowers record-level coverage (approximately 95.2-95.5%), while all attack types (6/6 CIC and 9/9 UNSW) remain present at every rate because of stratified selection. Rates of 20-30% yield marginal coverage gains (approximately 1%) with substantially less reduction. The 10% threshold therefore represents the most efficient operating point on the reduction-coverage trade-off curve.

In contrast, template hashing retained almost every attack entry ($> 99\%$) and reduced data size by about two-thirds by consolidating structurally identical flows. It maintains contextual information but adds higher CPU overhead because hashing operations and pattern comparisons require additional processing.

Field pruning achieved a moderate reduction of approximately 55%, retained full coverage, and improved processing latency; therefore, it suits forensic pipelines that still need field-level detail. Filter-based selection imposed minimal computational overhead but depended on known benign labels, which limits its practicality in unlabeled production environments.

In summary, sampling provides the highest space savings, hashing retains detailed context, pruning balances resource reduction with context retention, and filtering works best with pre-labeled data. The choice of reduction technique should therefore align with organizational objectives, such as sampling for storage-limited pipelines, hashing for correlation fi-

delity, or pruning for investigative use cases.

Table 8: Summary of Trade-offs and Recommended Use Cases.

Profile	Strength / Benefit	Limitation / Risk	Recommended SOC Use Case
Filter-only	Simplest approach; removes known benign traffic quickly; zero additional CPU cost	Requires labeled data or predefined rules; not adaptive to unseen threats	Labeled datasets, rule-based pipelines, or pre-ingest filtering for known traffic
Field Pruning	Retains full event count; reduces payload size by $\approx 55-58\%$; improves query latency	Slight loss of contextual fields may limit deep forensic analysis	Forensic or compliance pipelines where field-level detail is still needed but storage is constrained
Sampling (10%)	Highest efficiency ($\approx 90\%$ reduction); greatly improves throughput ($\sim 7.4k$ EPS); lightweight	May omit rare or low-frequency attacks; dependent on dataset redundancy	Large-scale MSSP or SIEM environments prioritizing storage and bandwidth efficiency
Template Hashing	Maintains $\approx 99\%$ attack coverage; consolidates redundant patterns; preserves structure	Moderate CPU overhead due to hashing; less effective if logs are already unique	SOC pipelines requiring structural deduplication or semantic compression
Combined (Pruning + Sampling)	Best overall balance: $\approx 90\%$ reduction, $\approx 98\%$ coverage; reduces both count and size	Compound configuration complexity; minor random loss of rare logs	Multi-tenant SOC seeking balanced efficiency and fidelity under resource limits

Note: The evaluation used Vector v0.41.0 at 5 kEPS on CIC-IDS2017 and UNSW-NB15 datasets. Throughput and coverage values reflect mean performance across three runs ($n = 3$).

5.2 Performance Insights

All profiles sustained more than 5 kEPS. Sampling achieved the best efficiency, whereas hashing balanced fidelity and cost. The 10% sampling profile delivered the highest effective throughput (7.30-7.40 kEPS) because it reduced the event count after processing. Template hashing incurred minor overhead, with throughput approximately 8-10% lower than field pruning and approximately 20-22% lower than sampling, yet it still remained above the 5.0 kEPS baseline. The consistency across profiles validates that Vector can embed log reduction natively

without external brokers or buffering layers. The 16-core system also confirmed throughput scalability, indicating suitability for multi-tenant SOC pipelines that process heterogeneous datasets. Table 4 summarizes the consistent throughput results across all reduction profiles.

5.3 Comparative Baseline and Practical Implications

For the baseline comparison, the experiment deployed Filebeat 7.12.1 under the same environment as the proposed Vector-based pipeline. The configuration used an equivalent input rate of 5,000 events per second (EPS) and the same output sink (ClickHouse) to ensure direct comparability. Filebeat kept all transformations and buffering features in their default state, emulating a traditional store-all log-forwarding approach without reduction.

Table 9: Baseline (Filebeat) vs. Proposed Vector-Based Profiles under 5 kEPS Condition.

Profile	Reduction (Count %)	Coverage (%)	Throughput (EPS)	CPU Usage (%)	Memory (MB)	Outbound Bytes (GB/h)
Filebeat (Baseline)	0.0	100.0	5,000	58.2	620	36.0
Filter-only	38.2	100.0	6,100	46.0	540	22.5
Field Pruning	0.0	99.5	6,300	44.7	530	18.1
Sampling (10%)	89.8	97.8	7,400	41.3	480	3.9
Template Hashing	65.2	99.2	5,800	49.5	570	12.8
Combined	90.5	98.1	7,200	42.0	490	3.6

Note: Metrics represent averages from three independent runs. Collector-level container metrics captured CPU and memory. Outbound bytes denote the effective data volume transmitted to ClickHouse per hour.

Table 9 reports throughput in EPS; Table 3 provides the corresponding kEPS values.

Table 9 compares the baseline Filebeat pipeline with the proposed Vector profiles. The Vector profiles reduced CPU usage by up to 30% while transmitting 10 times less data. Although Filebeat achieved 100% coverage, it produced approximately 10 times more outbound data and consumed 45-60% more CPU resources under the same setup. The analysis collected CPU utilization, memory footprint, throughput, and outbound bytes per hour at both the collector and sink layers. This baseline provides a practical reference for understanding the resource overhead and bandwidth cost of unfiltered ingestion.

As expected, the Filebeat pipeline achieved near-100% coverage but required significantly higher resources. Relative to the reduction profiles, Filebeat maintained similar ingestion throughput (approximately 5 kEPS) yet produced roughly 10 times more outbound data and incurred 45-60% higher CPU usage under identical hardware. These results show that simple reduction methods, such as sampling or

pruning, can reduce storage requirements and improve processing speed without losing critical alerts.

From a practical standpoint, MSSP or SOC operators could integrate Vector-based pruning or sampling directly within log forwarders to reduce upstream bandwidth without modifying SIEM dashboards or detection logic. This approach helps teams reduce costs immediately while maintaining visibility and existing detection logic.

5.4 Limitations

Future research should consider unlabeled SOC environments using unsupervised or semi-supervised coverage estimation, because labeled data may not be available in real SOC operations. The evaluated datasets (CIC-IDS2017 and UNSW-NB15) provide predefined labels for each flow, enabling accurate measurement of retained malicious instances. However, production log environments typically lack such labeled data, so researchers cannot directly quantify the coverage of unseen attacks.

In practice, label-free deployments rely on rule-based heuristics such as dropping high-volume DNS noise, low-severity firewall denies, or specific noisy ports. A simple port-based filter that drops ports 80, 443, 53, and 0 yields 75.0% reduction on CIC-IDS2017 but only 23.8% coverage. This result confirms that pure port filtering is overly aggressive and needs severity-aware or IOC-based rules. Future evaluations should incorporate anomaly-based selection for label-free environments.

Relying on labeled datasets may unintentionally lead to label leakage into the evaluation process, where knowledge of attack categories implicitly influences profile design and evaluation metrics. The study mitigated this risk by isolating labeling logic from reduction configuration. Future studies should therefore consider label-free or semi-supervised estimation techniques to approximate coverage under operational conditions.

The experiments used controlled replay rates and static hardware resources, which may differ from dynamic, multi-tenant SOC deployments. Therefore, researchers should interpret absolute throughput or latency values cautiously when extrapolating to large-scale production. Nonetheless, the comparative trends observed across profiles remain valid and can inform the design of adaptive log-reduction strategies.

Although the experiments used controlled containerized environments, external factors such as network jitter and log-source diversity might influence real-world reproducibility. Future work will extend validation across distributed nodes and varied SOC conditions.

5.5 Future Work

Future extensions include integrating distributed Vector clusters to evaluate scaling behavior under multi-source log ingestion. Another direction involves adaptive, learning-based reduction that responds to workload changes while reinforcement-learning agents balance storage savings and analytic utility. A multi-tenant SIEM/MSSP deployment could also enable per-tenant optimization, ensure fairness and compliance, and reduce ingestion cost. Expanding benchmark datasets to include Windows Event Logs, firewall telemetry, and M365 audit trails will further validate generalizability across enterprise environments. These future directions will extend the Vector-based benchmarking framework toward adaptive, large-scale, and multi-tenant observability pipelines.

6. CONCLUSION

This study evaluated five Vector-based log-reduction profiles: filtering, field pruning, sampling, template hashing, and a combined pruning + sampling profile. It measured their impact on volume, throughput, and coverage at the ClickHouse sink. Experiments on two public intrusion-detection datasets (CIC-IDS2017 and UNSW-NB15) showed that properly configured log reduction can preserve detection fidelity while reducing volume. Compared with the Filebeat reference setup, the sampling profile handled 45% more EPS, reduced outbound bytes by about 80%, and still retained approximately 98% of attack logs with approximately 90% volume reduction. Template hashing preserved high accuracy with moderate CPU use, whereas field pruning balanced resource savings and context retention.

The results indicate that SOC pipelines can exclude many duplicate events without reducing analytical accuracy, while still retaining key security-relevant evidence. The trade-off analysis suggests aligning reduction methods with organizational priorities and operational constraints, such as sampling for storage efficiency, hashing for fidelity, and filter-based selection for strict compliance or forensic requirements.

All tests used the same setup and ran three times to ensure consistent results. However, the evaluation relied on labeled datasets, which may not fully represent real-world log distributions in unlabeled environments. Future work will design adaptive, label-free reductions and test dynamic sampling guided by anomaly feedback, as well as integration with multi-tenant SOC pipelines.

Public configurations and scripts enable SOC engineers and MSSP architects to reuse this benchmark as a starting point for optimizing log pipelines under constrained resources while preserving threat-detection effectiveness. These results provide a reproducible foundation for future research in SOC log optimization and adaptive log reduction.

All experiment scripts and configurations are publicly available for reproducibility at <https://github.com/nutthakorn7/vector-log-reduction-benchmark>.

ETHICS AND AI USE DISCLOSURE

This study used only publicly available datasets (CIC-IDS2017 and UNSW-NB15). The study did not involve human subjects, personal data, or sensitive organizational logs. ChatGPT was used only to support language editing; the author independently performed the experimental design and analysis.

ACKNOWLEDGEMENT

The author thanks the supporting institution for providing research resources.

AUTHOR CONTRIBUTIONS

Conceptualization, N.C.; methodology, N.C.; software, N.C.; validation, N.C.; formal analysis, N.C.; investigation, N.C.; data curation, N.C.; writing—original draft preparation, N.C.; writing—review and editing, N.C.; visualization, N.C.; supervision, N.C.; funding acquisition, N.C. The author has read and agreed to the published version of the manuscript.

References

- [1] R. Marty, *Applied Security Visualization*, Addison-Wesley Professional, Upper Saddle River, NJ, USA, 2008.
- [2] K. Kent and M. P. Souppaya, "Guide to Computer Security Log Management," NIST Special Publication 800-92, National Institute of Standards and Technology, Gaithersburg, MD, USA, 2006.
- [3] G. González-Granadillo, S. González-Zarzosa and R. Diaz, "Security information and event management (SIEM): Analysis, trends, and usage in critical infrastructures," *Sensors*, vol. 21, no. 14:4759, 2021.
- [4] M. Landauer, F. Skopik, M. Wurzenberger and A. Rauber, "System log clustering approaches for cyber security applications: A survey," *Computers & Security*, vol. 92, no. 101739, 2020.
- [5] Vector, "Vector: Open-source observability data pipeline," [Online]. Available: <https://vector.dev>. [Accessed: 08-Jun-2026].
- [6] M. Di Mauro, G. Galatro, G. Fortino and A. Liotta, "Supervised feature selection techniques in network intrusion detection: A critical review," *Engineering Applications of Artificial Intelligence*, vol. 101, no. 104216, 2021.
- [7] M. Du, F. Li, G. Zheng and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," *Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS)*, Dallas, TX, USA, pp. 1285-1298, 2017.
- [8] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi and K. Tzoumas, "Apache Flink: Stream and batch processing in a single engine," *IEEE Data Engineering Bulletin*, vol. 38, no. 4, pp. 28-38, 2015.
- [9] T. Akidau *et al.*, "The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing," *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1792-1803, 2015.
- [10] J. Paulo and J. Pereira, "A survey and classification of storage deduplication systems," *ACM Computing Surveys*, vol. 47, no. 1:11, pp. 1-30, 2014.
- [11] W. Xia, H. Jiang, D. Feng, F. Douglass, P. Shilane, Y. Hua, M. Fu, Y. Zhang and Y. Zhou, "A comprehensive study of the past, present, and future of data deduplication," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1681-1710, 2016.
- [12] T. Kalamatianos, K. Kontogiannis and P. Matthews, "Domain independent event analysis for log data reduction," *Proceedings of IEEE 36th Annual Computer Software and Applications Conference (COMPSAC)*, Izmir, Turkey, pp. 225-232, 2012.
- [13] S. He, P. He, Z. Chen, T. Yang, Y. Su and M. R. Lyu, "A survey on automated log analysis for reliability engineering," *ACM Computing Surveys*, vol. 54, no. 6:130, pp. 1-37, 2021.
- [14] T. Zhang, H. Qiu, G. Castellano, M. Rifai, C. S. Chen and F. Pianese, "System log parsing: A survey," *arXiv preprint arXiv:2212.14277*, 2022.
- [15] N. Duffield, "Sampling for passive Internet measurement: A review," *Statistical Science*, vol. 19, No. 3, pp. 472-498, 2004.
- [16] I. Sharafaldin, A. H. Lashkari and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *Proceedings of 4th International Conference on Information Systems Security and Privacy (ICISSP)*, Funchal, Portugal, pp. 108-116, 2018.
- [17] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," *Proceedings of Military Communications and Information Systems Conference (MilCIS)*, Canberra, ACT, Australia, pp. 1-6, 2015.
- [18] I. Kotenko, D. Gaifulina and I. Zelichenok, "Systematic literature review of security event correlation methods," *IEEE Access*, vol. 10, pp. 43387-43420, 2022.
- [19] P. He, J. Zhu, Z. Zheng and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," *Proceedings of IEEE International Conference on Web Services (ICWS)*, Honolulu, HI,

- USA, pp. 33-40, 2017.
- [20] A. Pescapè, D. Rossi, D. Tammaro and S. Valenti, “On the impact of sampling on traffic monitoring and analysis,” *Proceedings of 22nd International Teletraffic Congress (ITC 22)*, Amsterdam, Netherlands, pp. 1-8, 2010.
- [21] M. Sedki, A. Hamou-Lhadj and O. Ait-Mohamed, “AWSOM-LP: An effective log parsing technique using pattern recognition and frequency analysis,” *arXiv preprint arXiv:2110.15473*, 2021.
- [22] M. Du and F. Li, “Spell: Streaming parsing of system event logs,” *Proceedings of IEEE 16th International Conference on Data Mining (ICDM)*, Barcelona, Spain, pp. 859-864, 2016.
- [23] S.-W. Huang, X. Wu and H. Li, “LogLSHD: Fast log parsing with locality-sensitive hashing and dynamic time warping,” *Proceedings of 21st International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE)*, Trondheim, Norway, pp. 11-20, 2025.
- [24] [24] B. Gedik, K.-L. Wu, P. S. Yu and L. Liu, “Adaptive load shedding for windowed stream joins,” *Proceedings of 14th ACM International Conference on Information and Knowledge Management (CIKM)*, Bremen, Germany, pp. 171-178, 2005.



Nutthakorn Chalaemwongwan is a lecturer in Computer Engineering at KOSEN-KMITL, Thailand. His research interests include cybersecurity, SIEM/XDR platforms, AI-driven SOC automation, cloud security, and threat intelligence. He is actively involved in applied research and industry collaboration projects related to security operations, multi-tenant log analytics, and AI-enhanced cyber defense systems. His recent works focus on reinforcement learning for autonomous SOC operations and scalable security analytics architectures.