

Performance Evaluation of Snort3 and Suricata: A Comparative Study

Ahmed Rabad^{1*}, Abdulmajid Al-Mqdashi¹, Fuaad Abdulrazzak¹

¹Faculty of Computers and Informatics, Tamar University, Yemen.

*Corresponding author. E-mail: ahmed.rabad@tu.edu.ye;

Contributing authors: almqdshi@tu.edu.ye;
fuaad.abdulrazzak@tu.edu.ye

Abstract

Network Intrusion Detection Systems (NIDS) are crucial for safeguarding modern networks, with Snort and Suricata being the most widely used open-source solutions. Most comparative studies focus on Snort2, whose single-threaded design is outdated compared to the redesigned Snort3. Recent comparisons between Snort3 and Suricata have produced inconsistent results due to variations in rulesets, traffic conditions, and experimental setups, leaving their relative efficiency and scalability unclear.

This study addresses this gap by providing a controlled and reproducible performance evaluation of the latest stable versions of Snort3 (v3.9.5.0) and Suricata (v8.0.1) under identical test conditions. Using the CIC-IDS2017 dataset, traffic was replayed at rates ranging from 100 Mbps to 1 Gbps, with rulesets varying from 10,000 to 50,000. Three performance metrics were examined: CPU utilization, memory consumption, and packet drop rate.

The results indicate that Suricata consistently achieved lower CPU usage, maintained a modest memory footprint, and sustained zero packet loss across all traffic loads and ruleset sizes. Snort3, in its default mode, exhibited higher CPU usage and began dropping packets at speeds exceeding 500 Mbps. Moreover, Snort3 experienced packet drops when the rulesets surpassed 30,000 rules, whereas Suricata remained resilient.

Overall, the findings reveal a clear trade-off: Snort3 offers greater efficiency when sufficient memory resources are available, whereas Suricata provides more consistent scalability and reliability, making it preferable for high-throughput and rule-intensive deployments.

Keywords: Snort3, Suricata, Network Intrusion Detection System, NIDS

DOI:

<https://doi.org/10.59421/vs44z864>



THIS WORK IS LICENSED UNDER A
CREATIVE COMMONS
ATTRIBUTION 4.0 INTERNATIONAL
LICENSE.

1 Introduction

The exponential growth of digital communication and increasing sophistication of cyber threats have made network security a cornerstone of modern information systems [1]. Organizations of all sizes rely heavily on security mechanisms that can detect, prevent, and respond to malicious activities in real time [2]. Within this security landscape, Network Intrusion Detection Systems (NIDS) play a pivotal role in analyzing network traffic and generating alerts when suspicious or malicious patterns are identified [3].

Among the wide range of available NIDS solutions, Snort and Suricata are the most prominent open-source tools [4]. Their popularity stems from their robust rule-based detection capabilities, active development communities, and integration into enterprise-grade security systems [5].

Snort, originally developed in 1998 by Martin Roesch from Sourcefire and now owned by Cisco, which acquired Sourcefire in 2013, gained recognition for its lightweight architecture, extensive community rulesets, and widespread adoption in both academic and industrial settings [6]. However, earlier versions of Snort (i.e., Snort2) were constrained by a single-threaded design, which limited their scalability in high-throughput environments. To address this, Snort3 (also known as Snort++), released as a complete redesign, introduced a modular architecture, native support for multi-threading, and a Lua-based configuration system, alongside optimizations such as hyperscan integration for accelerated pattern matching [7].

Suricata is another leading open-source NIDS released by the Open Information Security Foundation (OISF) in 2010 [6]. It was primarily started in response to the growing use of multi-cores in computers and to compensate for what Snort was missing. It shares many features with Snort while offering additional advanced capabilities, including automatic protocol detection, file extraction, logging HTTP requests, DNS queries, and TLS certificates. It has a multi-threaded engine and implements Lua scripting to detect things rules cannot, as well as advanced outputs to allow for easy integration with many logging tools [8].

Despite their widespread adoption, comparisons between Snort and Suricata have often focused on the older versions of Snort. Although influential, Snort2 is constrained by its single-threaded architecture, which limits its capacity to process high-speed traffic. This constraint renders many earlier comparative studies less relevant to modern high-throughput environments. The improvements in Snort3 significantly enhanced its efficiency and scalability, necessitating fresh comparisons with Suricata.

Recent research has begun to examine Snort3 in relation to Suricata; however, the findings have been inconsistent. Some studies highlight Snort3's improvements in memory efficiency and CPU utilization, whereas others emphasize Suricata's continued advantage in throughput and attack detection under multi-threaded loads. One of the primary challenges arises from the heterogeneity of experimental designs; differences in traffic generation, datasets, rulesets, and system configurations lead to divergent results. This fragmentation makes it difficult to draw firm conclusions regarding which system performs better for modern workloads.

Given these challenges, there is a clear need for an up-to-date, controlled, and reproducible study that compares Snort3 and Suricata. Such research should evaluate both systems under

identical conditions using standardized metrics that capture the core aspects of performance and scalability. By focusing on resource utilization (CPU and memory) and packet drop rates, researchers and practitioners can better assess the trade-offs between the two systems and make more robust and evidence-based deployment decisions.

This study addresses this gap by conducting an updated performance comparison of the latest stable versions of Snort3 and Suricata. The evaluation was conducted under controlled experimental conditions using synchronized rulesets and standardized measurement criteria. By aligning experimental conditions, this study seeks to generate reliable insights into their relative strengths and weaknesses, thereby contributing to both academic research and operational best practices. The key contributions of this study are summarized as follows:

- The proposed benchmarking offers an up-to-date comparative evaluation of Snort3 and Suricata.
- Comprehensive performance metrics were used to evaluate the CPU and memory usage and packet loss under controlled testbed conditions.
- Scalability analysis evaluates how ruleset complexity and traffic volume affect system performance, highlighting the practical trade-offs.

2 Objectives

The primary objective of this study was to provide a comparative performance evaluation of Snort and Suricata. The specific aims are:

1. Resource Utilization: To measure and compare the CPU and memory usage of Snort and Suricata under varying traffic loads.
2. Packet Processing Efficiency: Assessing packet drop rates for both systems in a controlled environment.
3. Impact of Ruleset Complexity: Analyzing how performance changes as the number and complexity of detection rules grow.

3 Related Work

A substantial body of research has compared open-source intrusion detection systems, with Snort and Suricata emerging as the most prominent candidates. Early comparative studies [9, 10, 11] largely concentrated on Snort version 2 because of its dominance in the field for over two decades. Although informative, these studies are increasingly outdated because Snort2's single-threaded architecture cannot adequately represent the capabilities of modern intrusion detection in high-throughput environments. With the release of Snort3, which incorporates multi-threading, modular configuration, and advanced pattern matching, the basis for comparison has shifted, and more recent investigations have sought to evaluate its performance relative to Suricata.

Several studies have explored this updated comparison, producing results that are sometimes convergent and sometimes contradictory.

Waleed et al. [12] conducted one of the earliest assessments involving Snort3, examining Snort2, Snort3, Suricata, and Zeek under traffic conditions representative of small and medium-sized enterprises. Their experiments demonstrated that Suricata consistently outperformed both Snort variants in terms of throughput, efficiency, and packet drop rate, which is attributed to its

multithreaded architecture. Snort is considered more appropriate for low-traffic environments, whereas Zeek is recommended for network monitoring rather than prevention.

In contrast, Boukebous et al. [13] compared Snort3, Snort2, and Suricata in terms of resource consumption, packet drop, and alert loss. Their results indicated that Suricata remained the most performant NIDS, particularly in handling high-throughput scenarios. However, they emphasized that Snort3 had significantly improved memory usage and packet processing speed compared to Snort2, with memory consumption ranging from 3.8% to 6.2%, which is lower than Suricata's reported 6.1% to 8.4%. They concluded that Snort3, although still relatively new and less mature, has emerged as a serious competitor to Suricata and is expected to improve further as its development stabilizes.

Hoover [14] further complicated the comparative picture by reporting that Suricata generated two to three times more alerts than Snort3 when both systems were configured with default rulesets and exposed to identical malicious traffic. Although CPU utilization was similar across both systems, Suricata's broader detection capabilities came at the cost of greater memory consumption, highlighting the trade-off between detection breadth and resource efficiency.

Other researchers have adopted a broader architectural and contextual perspective. Ghazi et al. [15] emphasized Snort's strength in signature-based detection and its wide adoption in production environments but underscored Suricata's superior scalability and reduced packet loss under high-volume traffic. Their study framed the comparison as an architectural philosophy: Snort offers extensive rulesets and long-standing community trust, and Suricata excels in efficiency through multi-threading. Looking ahead, they argued that both systems must evolve toward integrating artificial intelligence and machine learning to handle emerging challenges, such as encrypted traffic and zero-day attacks.

Expanding the scope beyond typical detection comparisons, Wahyu et al. [16] investigated the performance of Snort, Suricata, Zeek, and other intrusion prevention tools under distributed denial-of-service (DDoS) attack scenarios. Their evaluation considered not only the CPU and memory utilization but also the quality of service (QoS) metrics, including delay, jitter, throughput, and packet loss. Interestingly, their findings suggested that Snort outperformed Suricata in terms of QoS during ICMP and SYN flood conditions, even though both systems maintained efficient resource utilization. This highlights the contextual nature of performance evaluations: Suricata may dominate throughput-heavy scenarios, whereas Snort may better preserve service quality under certain attack conditions.

Taken together, existing studies provide useful perspectives on Snort3 and Suricata but also demonstrate significant inconsistencies in the results. These inconsistencies stem from differences in traffic generation, experimental setups, and NIDS configurations, as well as reliance on older versions, such as Snort3 v3.1.x or Suricata v6.x. Given the continuous improvements in both engines, including threading optimizations, performance tuning, and more efficient rule processing, earlier findings may no longer reflect their current capabilities.

This motivated the present study, which performs an updated and controlled performance comparison of the latest stable releases of Snort3 and Suricata using harmonized configurations and standardized traffic conditions. By focusing on CPU utilization, memory consumption, and packet drop rates, this study provides reliable, reproducible, and up-to-date insights into their relative strengths and trade-offs.

4 Methodology

This study employed an experimental research methodology to evaluate the performance of Snort3 and Suricata under controlled and reproducible conditions. The aim was to generate empirical evidence that highlights their relative strengths and limitations when subjected to identical traffic loads.

4.1 Experiment Setup

The experimental setup was designed to replicate real-world network conditions, while ensuring consistency and repeatability. To achieve this, a virtualized environment was constructed using VMware Workstation 16 Pro running on a laptop equipped with an Intel Core i7-8650U CPU, 32 GB of RAM, and a 1 TB SSD.

The testbed network, illustrated in Figure 1, consisted of three Virtual Machines (VMs) connected to a virtual switch provisioned with a 1 Gbps link, enabling traffic flows comparable to typical enterprise network speeds.

Each VM was allocated sufficient resources to ensure optimal performance during the experiments. Table 1 summarizes the specifications and roles of the main VMs used in this study.

Although a virtualized setup cannot fully replicate the performance of high-end multi-gigabit physical appliances, it provides a reliable environment for controlled benchmarking, flexibility, and ease of deployment, which aligns with the objectives of this study.

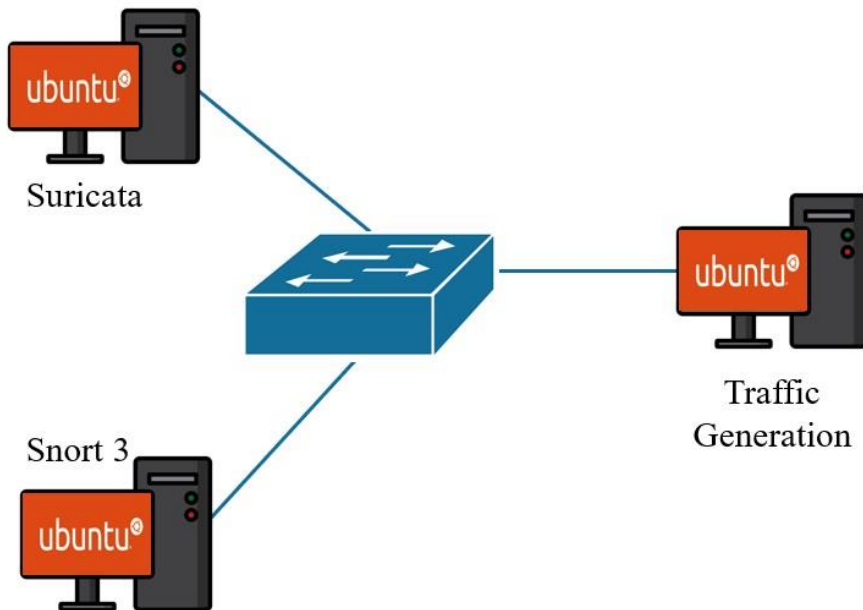


Figure 1: Testbed network.

Table 1: Network component specifications.

Machines/OS	Hardware Description	Role	Tools used
Ubuntu 22.04	4 CPU Cores, 8 GB RAM, 40 GB Hard drive	NIDS	Snort3 v(3.9.5.0)
Ubuntu 22.04	4 CPU Cores, 8 GB RAM, 40 GB Hard drive	NIDS	Suricata v(8.0.1)
Ubuntu 22.04	2 CPU Cores, 4 GB RAM, 20 GB Hard drive	Traffic Generator	Tcpreplay

4.2 Rulesets and Configurations

Both NIDSs were configured in NIDS mode, following the official documentation and deployment best practices [17, 8]. The latest stable versions were selected to reflect current capabilities: Snort3 v3.9.5.0 and Suricata v8.0.1.

- Suricata was configured using the Emerging Threats (ET) Open ruleset [18], a widely supported and up-to-date ruleset aligned with Suricata's multi-threaded architecture.
- Since Snort3 does not directly support ET Open, the Talos LightSPD ruleset was used instead. This ruleset is designed to work seamlessly with Snort3's modular setup and is recommended by Cisco's Snort team, with downloads available on their website.

To ensure fairness, the ruleset size was standardized across both engines. Five experimental groups were created with 10,000, 20,000, 30,000, 40,000, and 50,000 signatures to evaluate the scalability trends.

The Snort3 configurations included two scenarios:

1. Default mode: Using libpcap for packet acquisition and ac-bnfa for pattern matching.
2. Optimized mode: Using afpacket with hyperscan, which provides significant pattern-matching acceleration on modern CPUs.

Suricata was tested in its default multi-threaded configuration using afpacket and hyperscan (enabled by default). All logging, alerting, and output settings were standardized to ensure comparable activities across the systems.

Network-level optimizations, such as Generic Receive Offload (GRO) and Large Receive Offload (LRO), were manually disabled to prevent hardware-assisted aggregation from skewing the performance results.

4.3 *Traffic Generation and Dataset*

For this experiment, we utilized the CIC-IDS2017 dataset, a widely recognized and labeled benchmark from the Canadian Institute for Cybersecurity (CIC), specifically designed for evaluating IDS [19]. The dataset employs novel methods, such as the B-Profile system, to generate realistic benign background traffic that models abstract human behavior, thereby ensuring ecological validity.

The full CIC-IDS2017 dataset encompasses five days of network traffic within a simulated infrastructure of 25 machines, featuring a variety of attacks, including brute force (FTP, SSH), web attacks, botnet, and DDoS.

To maintain a focused and computationally tractable experiment, given our hardware constraints, we selected data from July 6, 2017. This day contains a mix of normal traffic and a representative suite of attacks, namely, brute force, web, and infiltration attacks. The resulting PCAP file was approximately 7.7 GB in size.

This curated PCAP file was replayed in our test environment using tcpreplay [20]. The tool was configured to replay the traffic at controlled incremental rates (ranging from 100 to 1000 Mbps). This approach allows us to evaluate both the raw performance ceiling and graceful degradation of each system under load.

4.4 *Metrics and Data Collection*

The evaluation focused on four standardized performance metrics:

- CPU utilization: The average percentage of CPU resources consumed by the NIDS process during traffic processing, measured using the system monitoring utility (*top*).
- Memory consumption: The proportion of physical memory utilized by the NIDS process, which was also recorded via *top*.
- Packet drop rate: The ratio of dropped packets to the total number of packets received, derived from NIDS log files and validated against NIC-level counters.

All measurements were automated using shell scripts to ensure consistency and reproducibility. The outputs were logged in real time and aggregated for statistical analysis and visualization.

Each experiment was performed for a fixed duration of 180 s and repeated five times to account for variability. The results are reported as arithmetic means across repeated runs.

5 *Results*

This section presents the performance evaluation of Snort3 and Suricata based on the experimental methodology described earlier. The analysis focused on three key metrics: CPU utilization, memory consumption, and packet drop rates under varying traffic loads (100-1000 Mbps).

5.1 *CPU Utilization*

Figure 2 shows how the CPU usage increased as the throughput increased. Default Snort3 consistently consumed the most CPU, starting at 14.4% at 100 Mbps and rising steadily to approximately 25.5% at 1 Gbps. This higher utilization reflects the overhead of its default pattern-matching engine (ac-bnfa) and the single-threaded packet acquisition via libpcap.

In contrast, the optimized Snort3 (afpacket + hyperscan) demonstrated substantial improvements, with the CPU usage reduced by an average of 5-7% across all traffic levels. Hyperscan's vectorized pattern-matching capabilities significantly reduce the processing overhead, particularly under high-speed traffic.

Suricata maintained the lowest CPU utilization overall, starting at 6.9% at 100 Mbps and rising modestly to 16% at 1 Gbps. This efficiency stems from Suricata's multithreaded architecture, which distributes the workload across the available CPU cores, thereby reducing the per-core stress.

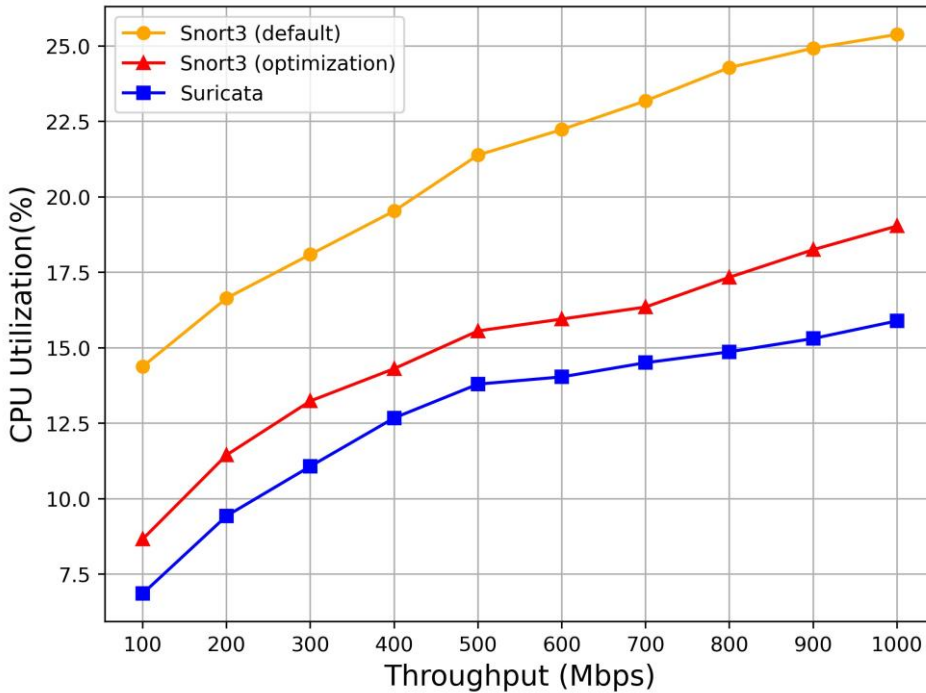


Figure 2: CPU utilization of Snort3 and Suricata under increasing throughput.

5.2 Memory Consumption

The memory usage remained stable across all configurations under varying traffic loads, as shown in Figure 3. Snort3 in default mode maintained a consistent memory footprint of 1.6% of the system memory across the tested throughput range (100-1000 Mbps). Suricata exhibited slightly higher but stable memory consumption of 2.0%, reflecting its multi-threaded architecture and integrated detection features.

In contrast, Snort3 in the optimized mode (with afpacket and hyperscan enabled) demonstrated a significant increase in memory usage, reaching 15.9% even at lower ruleset sizes. This substantial increase is attributable to the design of hyperscan, which precompiles and loads large signature databases into memory to accelerate pattern matching. Although this approach

reduces the CPU load, it introduces a pronounced memory overhead, which becomes more constraining as the ruleset size increases.

These results highlight a clear trade-off: optimization in Snort3 shifts the resource demand from the CPU to memory, whereas Suricata maintains a balanced and predictable resource profile. Suricata's moderate memory footprint, combined with its multi-threading efficiency, makes it particularly suitable for environments where memory availability is constrained or where consistent performance under scaling rulesets is required.

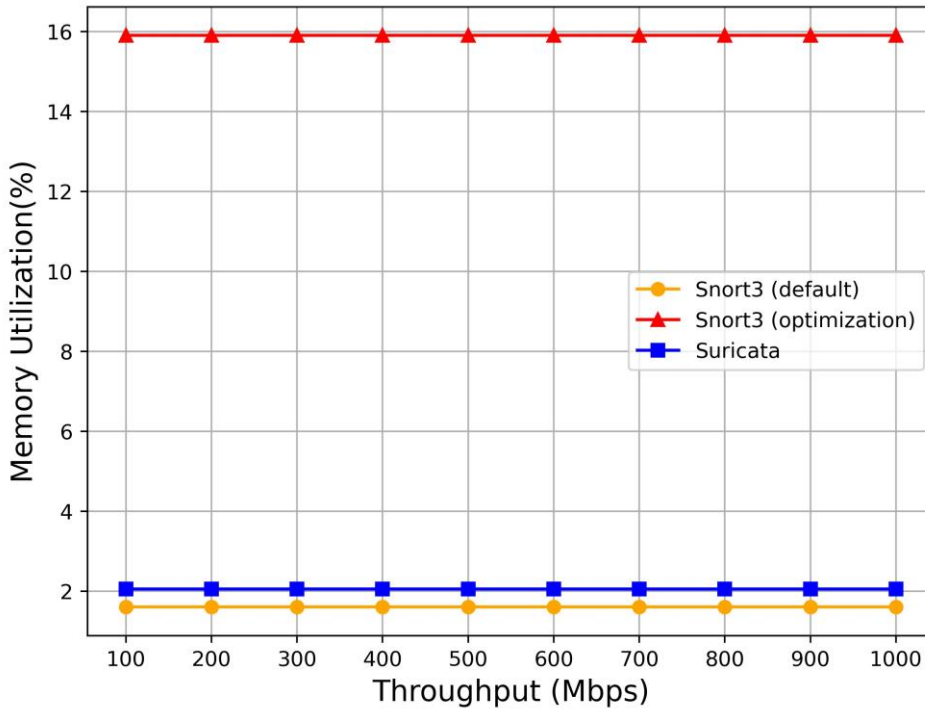


Figure 3: Memory utilization of Snort3 and Suricata under increasing throughput.

5.3 Packet Drop Rate

Packet retention is a critical metric for evaluating NIDS reliability because dropped packets may contain malicious activity and compromise detection coverage. Figure 4 illustrates how the packet loss varied with increasing throughput levels.

Default Snort3 began experiencing packet loss once the throughput exceeded 500 Mbps, reaching a 5.1% loss at 1 Gbps. This limitation is largely attributable to its reliance on libpcap for packet capture, which introduces a per-packet processing overhead and struggles under high traffic in the single-threaded mode.

The optimized Snort3 (using afpacket + hyperscan) eliminated packet loss entirely across all tested throughput levels. The afpacket capture module provides a higher packet capture

capacity than libpcap, whereas hyperscan accelerates pattern matching, reduces per-packet processing latency, and prevents queue overflow.

In contrast, Suricata maintained zero packet loss at all throughput levels, benefiting from its built-in multi-threaded packet acquisition and processing pipeline, which spreads the workload across CPU cores and minimizes capture bottlenecks.

These results demonstrate that the packet handling performance is highly dependent on the underlying capture mechanism and pattern-matching efficiency. Although the optimized Snort3 can match Suricata's loss-free performance under moderate loads, its default configuration remains susceptible to packet drops in high-throughput scenarios. Suricata's design ensures consistent packet retention, reinforcing its suitability for environments in which traffic integrity is paramount.

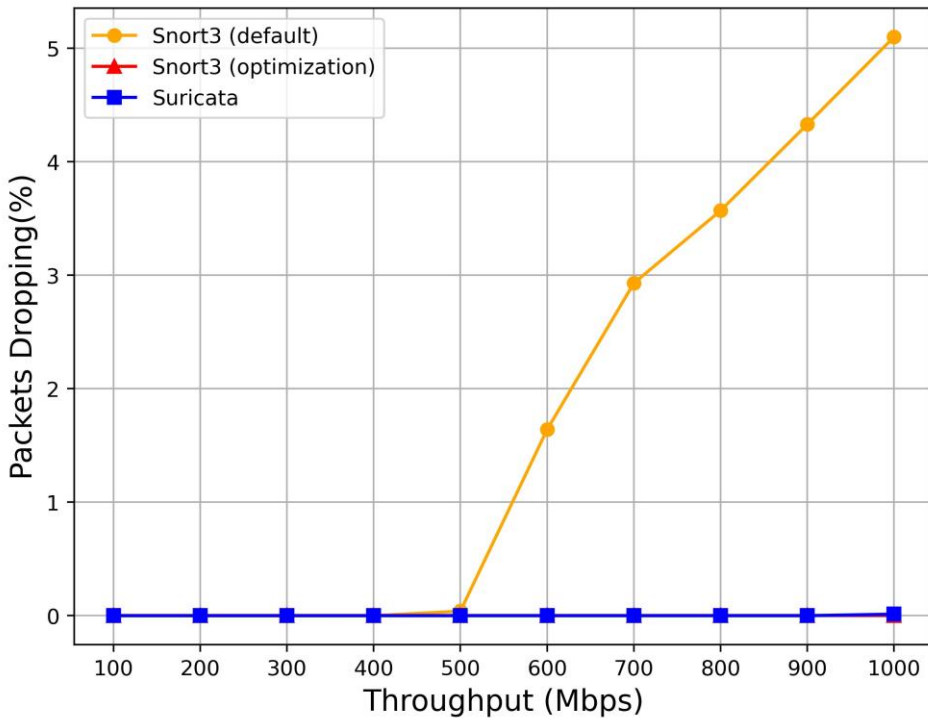


Figure 4: Packet drop rates of Snort3 and Suricata under increasing throughput.

5.4 Impact of Ruleset Size

To assess the effect of ruleset complexity on system performance, the throughput was fixed at 1000 Mbps, whereas the ruleset size was varied from 10,000 to 50,000 rules. Both Snort3 (in the optimized mode with afpacket + hyperscan) and Suricata were evaluated.

Figure 5 illustrates the impact of ruleset size on CPU utilization, memory consumption, and packet drops. The results reveal clear performance trends:

- CPU utilization: Both Snort3 and Suricata showed steady increases in CPU usage as the ruleset size grew. At the maximum size (50,000 rules), Snort3 reached nearly 30% CPU, while Suricata followed closely at approximately 27%.
- Memory consumption: Suricata consistently required far lower memory than Snort3 across all ruleset sizes. Suricata's memory usage rose modestly from 2% to 5%, whereas Snort3 consumed significantly more (16% at 10,000 rules, climbing above 21% at 50,000 rules).
- Packet drop rate: Suricata sustained zero packet loss across all configurations. Snort3 remained drop-free up to 20,000 rules but began incurring losses beyond 30,000 rules, reaching nearly 2% packet loss at 50,000 rules.

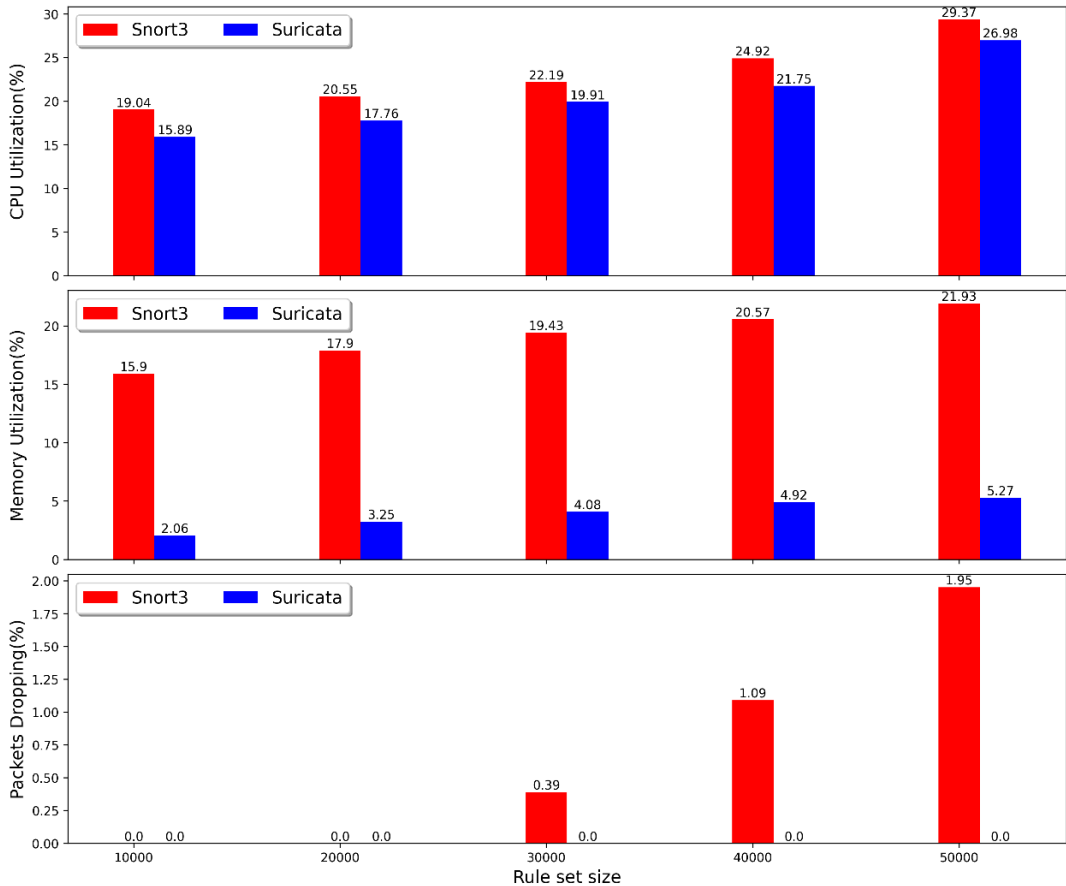


Figure 5: Performance of Snort3 and Suricata against ruleset size.

These findings suggest that while both NIDS scale in terms of CPU usage with larger rulesets, Suricata is more memory-efficient and resilient, maintaining full packet capture even at

50,000 rules. Snort3, although effective under moderate ruleset sizes, incurs higher memory costs and begins to lose packets when pushed beyond 30,000 rules.

6 Discussion

The experimental results highlight several important trade-offs between Snort3 and Suricata when deployed as NIDS.

CPU Utilization: The relationship between CPU utilization and the chosen pattern-matching algorithm emerged as a critical factor in both our results and those of prior studies. In our experiments, Suricata in its default configuration consumed less CPU than default Snort3, consistent with the general view of Suricata's efficient multi-threaded design (Boukebous et al. [13]).

However, this trend is not universal. Waleed et al. [12] reported that Suricata exhibited higher CPU utilization than Snort3 when all systems were tested using the Aho-Corasick (AC) algorithm. This apparent contradiction can be explained by algorithmic constraints: although the AC algorithm is robust, it is not optimized for CPU efficiency on modern hardware. Suricata's architecture is highly tuned for its own advanced multi-pattern matchers, but when restricted to the AC algorithm, it loses much of its efficiency advantage.

Our study further shows that Snort3's CPU performance improves substantially when configured with hyperscan, a high-throughput pattern-matching library optimized for modern processors. With hyperscan enabled, Snort3's CPU load decreased significantly, narrowing the gap with default Suricata.

Memory Consumption: A key point of divergence between Snort3 and Suricata lies in their memory usage. In our experiments, Suricata consistently maintained a modest and stable memory footprint, even as ruleset sizes increased to 50,000 entries. By contrast, Snort3 in optimized mode (afpacket with hyperscan) exhibited substantially higher memory consumption, exceeding 20% at larger rulesets, while default Snort3 consumed amounts closer to Suricata.

These results differ from those of Boukebous et al. [13], who reported that Snort3 achieved lower memory usage than Suricata, and from Hoover [14], who likewise observed that Suricata was more memory-intensive than Snort3 under default configurations. The discrepancy likely stems from variations in system configurations and experimental setups, as neither Boukebous et al. nor Hoover specifically tuned parameters such as pattern matching algorithms, rule compilation methods, or logging options.

Taken together, these findings suggest that while Suricata's baseline memory footprint may be somewhat higher under default configurations, the use of advanced optimizations in Snort3 (e.g., hyperscan) shifts the balance, making Snort3 considerably more memory demanding. This highlights the importance of reporting and analyzing configuration details, since they directly affect the comparability of performance studies.

Packet Drops: Packet retention is critical for NIDS reliability. Our results show that Suricata sustained zero packet loss across all traffic loads and ruleset sizes, reinforcing its resilience under stress. Optimized Snort3 also exhibited excellent packet-handling performance, with no drops observed up to 20,000 rules, but began dropping packets when the ruleset exceeded this threshold. Default Snort3 performed poorly at high throughput, with losses reaching 5% at 1

Gbps. These findings complement Wahyu et al. [16], though their configuration details are not specified, making direct performance alignment difficult.

Impact of Ruleset Size: The ruleset scalability experiment revealed that both systems experience gradual increases in CPU and memory usage as the number of signatures grows. However, while Suricata scaled smoothly with negligible packet drops, Snort3 encountered a clear threshold beyond 30,000 rules where packet losses became significant. Since previous studies [14, 13] did not specify their ruleset configurations, standardize the number of rules, or exhibit configuration variation, our results provide a clearer view of how scaling affects Snort3 and Suricata under controlled conditions.

7 Limitations

Although the study offers controlled and reproducible insights, several limitations should be acknowledged.

First, the experiments were conducted in a virtualized environment using VMware Workstation. While virtualization provides flexibility and isolation, it does not fully replicate the characteristics of physical network appliances. Virtual NIC behavior, shared CPU scheduling, and hypervisor overhead can influence packet timing and throughput, potentially affecting how each NIDS handles traffic under load.

Second, the entire testbed operated on a single physical host machine. Although each VM was assigned fixed CPU and memory resources, the underlying hardware was still shared. This differs from dedicated multi-node or hardware-accelerated deployments, which may exhibit different behavior due to factors such as hardware offloading, parallel processing pipelines, or multi-interface architectures.

Third, the ability of each NIDS under test to detect specific attacks is inherently influenced by whether corresponding rules are present in the default ruleset. Signature-based NIDS rely on predefined patterns to identify threats; therefore, missing or incomplete signatures can limit detection capability. For this reason, the study focused exclusively on performance-oriented metrics—CPU utilization, memory usage, packet drop rate, and ruleset scalability—and did not evaluate detection accuracy metrics such as precision, recall, or MCC.

These limitations provide important context for interpreting the results and highlight opportunities for future work involving dedicated hardware platforms, distributed deployments, and extended evaluation dimensions such as rule coverage and detection quality.

8 Conclusion

This study has presented a controlled comparative evaluation of Snort3 and Suricata, two of the most widely adopted open-source network intrusion detection systems. By analyzing CPU utilization, memory consumption, packet drop rates, and the impact of ruleset scalability under

controlled conditions, the research provides an updated and reliable performance benchmark for these tools.

The findings show that Suricata consistently demonstrated lower CPU utilization, stable memory usage, and zero packet loss, even at high throughput and large ruleset sizes. This resilience underscores its suitability for high-speed and large-scale deployments, such as in large enterprises. Snort3, by contrast, exhibited higher CPU utilization and packet drops in its default configuration, but when optimized with afpacket and hyperscan, it achieved substantial improvements in CPU efficiency and maintained loss-free performance at moderate ruleset sizes. These improvements, observed under the controlled experimental conditions described in the Methodology section, came at the cost of significantly higher memory consumption, and Snort3 began to incur packet loss once rulesets exceeded 30,000 entries.

Taken together, these results highlight a trade-off between efficiency and resilience. Snort3 is attractive in environments where configurability and CPU efficiency are valued and where adequate memory resources are available, making it suitable for small to medium-sized organizations or research settings. Suricata, on the other hand, is more reliable under demanding traffic conditions and extensive rulesets, which positions it as a stronger choice for large-scale or mission-critical deployments.

Future work should extend this analysis to higher-speed testbeds operating at 10-100 Gbps. In addition, evaluating detection accuracy using precision, recall, and MCC, as well as conducting long-term studies in real-world environments, will provide a more comprehensive picture of the operational trade-offs between Snort3 and Suricata. By addressing these directions, subsequent research can bridge the gap between controlled laboratory benchmarks and the practical realities of production network environments.

References

- [1] H. A. Salman and A. Alsajri, "The evolution of cybersecurity threats and strategies for effective protection. A review," *SHIFRA*, vol. 2023, pp. 73–85, 2023.
- [2] R. Adriko and J. R. Nurse, "Cybersecurity, cyber insurance and small-to-medium-sized enterprises: a systematic review," *Information & Computer Security*, vol. 32, no. 5, pp. 691–710, 2024.
- [3] O. H. Abdulganiyu, T. Ait Tchakoucht, and Y. K. Saheed, "A systematic literature review for network intrusion detection system (IDS)," *International Journal of Information Security*, vol. 22, no. 5, pp. 1125–1162, 2023.
- [4] S. Zaib, "An in-depth comparative analysis and evaluation of Snort and Suricata IDS/IPS functionality in pfSense," *Global Scientific Journal (GSJ)*, vol. 11, no. 8, 2023.
- [5] H. Asad, S. Adhikari, and I. Gashi, "A perspective retrospective analysis of diversity in signature-based open-source network intrusion detection systems," *International Journal of Information Security*, vol. 23, no. 2, pp. 1331–1346, 2024.
- [6] H. El-Ta, M. Fahhad, R. Abumlha, S. Showman, and Z. Saab, "A lightweight network intrusion detection system for SMEs," *International Journal of Network Security & Its Applications*, vol. 17, no. 3, 73–93, May 2025. doi: [10.5121/ijnsa.2025.17306](https://doi.org/10.5121/ijnsa.2025.17306).

- [7] C. Sheng, S. Zhang, Y. Liu, and H. Liao, "Regular expression implementation: a comprehensive review from software algorithms to hardware acceleration," Z. Xiong and Z. Ning, Eds. Guangzhou, China: SPIE, Aug. 2024, p. 49.
- [8] *Suricata user guide—suricata 8.0.1 documentation*, <https://docs.suricata.io/en/suricata-8.0.1/>, accessed: May 2025.
- [9] A. Gupta and L. S. Sharma, "Performance evaluation of snort and suricata intrusion detection systems on ubuntu server," in *Proceedings of ICRIC 2019*, 2020, pp. 811-821.
- [10] F. Alsakran, G. Bendiab, S. Shiaeles, and N. Kolokotronis, "Intrusion detection systems for smart home iot devices: Experimental comparison study," in *International Symposium on Security in Computing and Communication*, 2019, pp. 87-98.
- [11] W. Park and S. Ahn, "Performance comparison and detection analysis in snort and suricata environment," *Wireless Personal Communications*, vol. 94, pp. 241-252, 2017.
- [12] A. Waleed, A. F. Jamali, and A. Masood, "Which open-source ids? snort, suricata or zeek," *Computer Networks*, vol. 213, pp. 109-116, 2022.
- [13] A. A. E. Boukebous, M. I. Fettache, G. Bendiab, and S. Shiaeles, "A comparative analysis of snort 3 and suricata," in *2023 IEEE IAS Global Conference on Emerging Technologies (GlobConET)*. IEEE, 2023, pp. 1-6.
- [14] C. Hoover, "Comparative study of snort 3 and suricata intrusion detection systems," *Computer Science and Computer Engineering Undergraduate Honors Theses* Retrieved from <https://scholarworks.uark.edu/csceuht/10>, vol. 5, 2022.
- [15] D. S. Ghazi, H. S. Hamid, M. J. Zaiter, and A. S. G. Behadili, "Snort versus suricata in intrusion detection," *Iraqi Journal of Information and Communication Technology*, vol. 7, no. 2, pp. 73-88, 2024.
- [16] A. P. Wahyu, K. Fauziah, A. S. Nahrowi, M. N. Faiz, and A. W. Muhammad, "Strengthening network security: Evaluation of intrusion detection and prevention systems tools in networking systems," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 9, 2023.
- [17] *Snort setup guides for emerging threats prevention*, <https://www.snort.org/documents> , accessed: May 2025.
- [18] Proofpoint Inc., *Proofpoint emerging threats rules*, <https://rules.emergingthreats.net/>, 2025, accessed: September 2025.
- [19] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, January 2018.
- [20] Klassen, *Tcpreplay—pcap editing and replaying utilities*, <https://tcpreplay.appneta.com/>, accessed: July 2025.