Naif Arab University for Security Sciences
**Journal of Information Security and Cybercrimes Research**
مجلة بحوث أمن المعلومات والجرائم السيبرانية
https://journals.nauss.edu.sa/index.php/JISCR
JISCR

# Assessing the Effectiveness of Open-Source Network Intrusion Detection Systems for Small-to-Medium-Sized Enterprises

**Ghadi M. A. Alzahrani[1], Nizar H. Alsharif[1], and Moez Krichen[2,3,*]**
1Faculty of Computing and Information, Department of Computer Science, Al-Baha University, Al-Baha, Saudi Arabia
2Faculty of Computing and Information, Al-Baha University, Al-Baha, Saudi Arabia
3ReDCAD Laboratory; Sfax University, Sfax, Tunisia.

*Received 11 Dec. 2025; Accepted 27 Dec. 2025; Available Online 30 Dec. 2025*

## Abstract

Network security is a critical concern for small and medium-sized enterprises (SMEs), often lacking resources for comprehensive solutions. This study evaluates three open-source network intrusion detection systems (NIDS): Snort, Suricata, and Zeek, to assess their suitability for SMEs. Using a controlled, virtualized environment, we simulated realistic SME network conditions and subjected each NIDS to tests measuring their ability to handle high traffic volumes and various attack types, including DoS, malware, ransomware, and phishing. Results showed that Suricata consistently outperformed the others in scalability, resource efficiency, and detection accuracy, achieving high true positive rates while minimizing false positives, which is essential for reducing alert fatigue among SME users. Snort 3, optimized with afpacket and hyperscan, also demonstrated strong capabilities but required more resources, while Snort 2 struggled with high-volume traffic. Although Zeek is lightweight, it was less effective in signature-based detection but excelled in monitoring anomalies. This study provides insights to guide SMEs in selecting appropriate NIDS based on their specific requirements and emphasizes the need for ongoing optimization and further research in physical environments.

## I. Introduction

Technological advancements have significantly transformed business operations, with the internet enhancing communication, collaboration, e-commerce, and remote work for companies. However, this reliance on interconnected networks has also amplified the risks of cyberattacks [1]. The cybersecurity threat landscape continues to evolve, becoming increasingly complex and diverse as various malicious actors employ sophisticated strategies to breach networks and systems. Phishing attacks, ransomware, and other advanced persistent threats have become commonplace, posing significant challenges for organizations of all sizes.

Small and medium enterprises (SMEs) are particularly vulnerable to these cyber threats. Unlike larger organizations with specialized security teams

**Keywords:** Network intrusion detection, open-source, small-to-medium-sized enterprises

Production and hosting by NAUSS

* Corresponding Author: Moez Krichen
Email: m.krichen@redcad.org
doi: 10.26735/EWOS4922

and substantial financial resources, SMEs often struggle to implement comprehensive cybersecurity measures [2]. These enterprises typically lack the necessary expertise to maintain robust security infrastructures and often view cybersecurity as an expense rather than an essential investment. While basic protective measures such as firewalls and antivirus programs offer some defense, they may not effectively detect advanced threats, leaving SMEs exposed to potential breaches [3].

Network intrusion detection systems (NIDS) have emerged as a vital component of cybersecurity, enhancing an organization's security posture by monitoring network traffic in real-time to identify potential threats [4]. By actively analyzing incoming and outgoing traffic, NIDS can detect malicious activities and provide alerts, enabling organizations to respond swiftly to security incidents.

Although commercial NIDS products offer robust protection, they often come with high financial costs, creating a security gap for SMEs with limited IT budgets [5]. The financial burden associated with licensing and maintenance can be prohibitive for many SMEs, leading them to overlook critical cybersecurity measures. Open-source NIDS, such as Snort, Suricata, and Zeek, present a cost-effective alternative, allowing SMEs to leverage advanced intrusion detection capabilities without incurring licensing fees [6]. These tools also foster a communitydriven approach to security, where users can benefit from shared intelligence and constant updates.

However, selecting the most suitable open-source NIDS for an SME involves careful consideration of various factors, including performance, detection capabilities, and ease of management. Each NIDS has its strengths and weaknesses, and understanding how these tools perform in real-world scenarios is crucial for making informed decisions. Furthermore, SMEs must assess their specific needs and resource constraints to identify the best fit for their operational environments.

This research aims to evaluate three widely used open-source NIDS—Snort, Suricata, and Zeek—enabling SMEs to make informed choices based on their specific needs and resource constraints. The comparative analysis will provide valuable insights by examining the features of each NIDS, focusing on ease of deployment, performance under varying network conditions, and detection efficacy against prevalent cyberattacks targeting SMEs.

This work contributes significantly to the field of NIDS by offering:

- A comprehensive review of existing NIDS tools, highlighting their strengths and weaknesses in various deployment scenarios.
- A standardized methodology for evaluating the performance of Snort, Suricata, and Zeek, providing a framework for future research.
- Empirical analysis of performance metrics under varying network conditions, offering insights into operational efficiency.
- An assessment of detection capabilities against common cyber threats, evaluating effectiveness for SMEs.
- Practical recommendations for SMEs on selecting and implementing NIDS solutions tailored to their technical expertise and resource availability.

This paper is organized into five sections to provide a comprehensive overview of the research. Section II reviews the background and related work, establishing context and significance within the existing literature. Section III details the methodology and experiments conducted, outlining the procedures used to evaluate the NIDS under consideration. Following this, Section IV presents performance evaluation results, analyzing the effectiveness and efficiency of each system. Section V assesses the detection capabilities against various cyber threats, highlighting strengths and limitations. Finally, Section VI concludes with recommendations and directions for future research, summarizing key findings and implications.

## II. Background and Related Work

SMEs, despite being the backbone of many economies, face significant vulnerabilities due to limited resources, exposing them to various cyberattacks. Key challenges include:

- Limited IT Budget and Expertise: SMEs

typically operate with constrained budgets and smaller IT teams, making it challenging to establish robust cybersecurity measures.

- Increased Reliance on Third-Party Vendors: Heavy dependence on third-party services (e.g., cloud computing and software solutions) can introduce vulnerabilities that compromise the SME's overall security.
- Cloud Security Concerns: While cloud services offer flexibility, they also present security risks. SMEs may lack the resources to configure and manage cloud security effectively, increasing their exposure to breaches.
- Prevalent Social Engineering and Phishing Attacks: Cybercriminals frequently target SMEs through social engineering and phishing, exploiting potential gaps in employee security awareness to gain sensitive information.
- Limited Security Awareness Training: Budget constraints often hinder SMEs from providing comprehensive security training, leaving employees ill-equipped to recognize and respond to cyber threats.

SMEs are increasingly targeted by cyberattacks due to their perceived vulnerabilities, including limited cybersecurity resources and outdated practices. The most prevalent cyberattack threats that SMEs face are:

- General Malware: This includes viruses, worms, trojans, spyware, and adware designed to infiltrate and damage systems. Malware can compromise sensitive data, disrupt operations, and incur substantial financial costs due to recovery and lost customer trust.
- Ransomware: A destructive form of malware that encrypts data, rendering it inaccessible until a ransom is paid, typically in cryptocurrency. SMEs are attractive targets due to often lacking sophisticated defenses, leading to operational downtime and high recovery costs.
- Phishing: A social engineering attack that deceives individuals into revealing sensitive information through fraudulent emails or websites. SMEs may be particularly vulnerable due to inadequate cybersecurity training, leading to unauthorized access and potential financial fraud.
- Web Application Attacks: These attacks exploit vulnerabilities in online services, such as e-commerce platforms. Common methods include SQL injection and cross-site scripting (XSS), which can compromise sensitive information and disrupt business operations.
- Denial-of-Service (DoS) Attacks: DoS attacks aim to overwhelm services with excessive traffic, making them unavailable to legitimate users. SMEs, especially those dependent on online interactions, can suffer significant losses due to downtime, including lost sales and damage to brand reputation.

Intrusion Detection Systems (IDS) are software applications that monitor network traffic for potentially malicious activities. IDS can operate in two main modes: alerting on suspicious activities (IDS) or actively blocking harmful traffic (Intrusion Prevention Systems, IPS). There are two primary categories of IDS: Network IDS (NIDS), which monitor traffic across an entire network, and Host IDS (HIDS), which focus on individual devices. IDS can also be classified by their detection methods: anomalybased systems that identify deviations from established traffic patterns and signature-based systems that trigger alerts based on known patterns of malicious activity [7]. This study focuses on open-source NIDS. Open-source NIDS provide flexible and customizable options for enhancing network security. Notable examples include Snort, which is a widely used NIDPS that operates on a signature-based model. It supports both IDS and IPS modes, allowing for traffic monitoring and alert generation. Snort's architecture includes packet capturing, decoding, normalization, detection, and output generation. Although it lacks a graphical user interface, visualization tools like Snorby can enhance its usability. The release of Snort 3 introduced multithreading, improving packet processing capabilities. Another notable system is Suricata, developed by the Open Information Security Foundation (OISF) as an IDS/IPS and network monitoring tool. Unlike Snort, Suricata uses a multi-threaded architecture, enabling

efficient packet processing. Its architecture comprises packet capture, decoding, detection, and output alert modules, allowing for concurrent detection. Zeek operates solely in IDS mode and is maintained by the Zeek Project. It features a scalable architecture that includes workers for log transmission to a manager, which processes events and applies detection rules. Zeek supports anomaly-based detection, setting it apart from Snort and Suricata, which focus on misuse detection. Despite its efficiency, Zeek has a limited number of default signatures, which may affect its adoption compared to Snort and Suricata [8].

The use of Network Intrusion Detection Systems (NIDS) is crucial for organizations to protect their networks and data from attacks. However, their performance can be significantly affected by hardware issues, leading to dropped packets and potential vulnerabilities [9]. Consequently, performance testing of NIDS has become a prominent area of research [10], [11]. Many studies have focused on open-source NIDS, particularly comparing the performance of Snort and Suricata. In [12], Snort and Suricata were analyzed on different platforms at traffic rates up to 2 Gbps, revealing that Suricata outperformed Snort on Linux, especially at high speeds, though Zeek was not included in the comparison. Research in [13] found that Snort outperformed Suricata in single-core setups, while Suricata excelled in multi-core environments, highlighting its scalability. A comparison in [14] indicated that Suricata had a lower packet drop rate and better performance than Snort but required more computational resources. The introduction of BSnort in [15], a modified Snort, showed improved performance against DoS attacks, but focused solely on Snort. Stress testing conducted in [16] revealed high packet drop rates for Snort under heavy traffic, leading to a proposed parallel architecture. In [17], Snort and Suricata were tested under stress, with results favoring Suricata. Additional studies [18] and [19] confirmed Suricata's superior performance, particularly under high traffic conditions, although neither included comparisons with Zeek. Various research efforts, including [20], consistently found that Suricata outperformed Snort in terms of processing higher loads, though Snort maintained better accuracy. A Recent study [21] evaluated both Snort and Suricata in virtualized environments, with findings indicating that Suricata generally outperformed Snort across various parameters. However, these evaluations often neglected Zeek and did not assess conditions relevant to small and medium-sized enterprises (SMEs). Research in [22] highlighted performance metrics during simulated DDoS attacks, showcasing Snort's effectiveness against ICMP floods and Suricata's against SYN floods, yet did not consider usability for non-technical SME staff. The study [23] demonstrated improvements in Snort 3 over Snort 2, particularly in memory management and reduced packet loss, but noted that Suricata remains more suitable for large networks.

In conclusion, there are notable gaps in the current knowledge base regarding the latest versions of Snort, Suricata, and Zeek, as well as new open-source signature rule sets, advancements in software and hardware technologies, and emerging attack methods. Additionally, comprehensive evaluations of these three prominent open-source solutions are lacking. This research aims to address these gaps by utilizing the most recent versions of Snort, Suricata, and Zeek, along with updated signature rule sets and new attack strategies to assess performance. In this context, our study focuses on evaluating open-source NIDS specifically in the context of SME networks, emphasizing the following aspects:

- Ease of Deployment and Management: Assessing the installation, configuration, and management simplicity for each IDS, considering the typical technical expertise of SME staff.
- Performance Evaluation under SME Network Conditions: Evaluating CPU, memory, and bandwidth usage under simulated network traffic patterns representative of typical SME activity to provide a realistic picture of resource demands.
- Detection Capability Assessment for Common SME Threats: Focusing on cyberattacks frequently encountered by SMEs, such as phishing attempts, malware downloads, ransomware, web application attacks, and DoS attacks.

By addressing these gaps, this research will offer practical guidance for SME network administrators in selecting opensource NIDS. The findings will assist in choosing an NIDS that balances ease of use, resource efficiency, and effective threat detection relevant to SME network environments.

## III. Methodology And Experiments

This research employs an experimental methodology to assess the effectiveness of three open-source NIDS: Snort (versions 2 and 3), Suricata, and Zeek, within the context of SMEs. It addresses three key aspects: ease of deployment and management, performance under varying network conditions, and detection accuracy for common SME cyber threats. A series of controlled experiments were conducted in a virtualized environment simulating realistic SME network conditions and attack scenarios. The selected NIDS were chosen for their widespread use, community support, and open-source nature, which eliminates licensing costs. These tools will collectively be referred to as NIDS Under Test (NUTs). The experimental design aimed to systematically evaluate the performance of the NUTs. Each scenario was crafted to address specific research questions, focusing on detection capability, resource utilization (CPU and memory), and packet drop rates. A diverse set of attack scenarios was selected to evaluate the robustness of each NUT, with varying network traffic intensity and volume. To ensure objective comparisons, each NUT was tested under identical conditions, with performance metrics monitored and recorded during simulations. The experiments were conducted 10 times to ensure reliability, and results were averaged to provide consistent findings.

The experiment setup is designed to create a controlled environment that closely mimics real-world network conditions while ensuring consistency and repeatability across all experiments. To achieve this, virtualization technology was utilized to create a virtualized environment where each experiment component could be isolated and managed independently. This approach offers flexibility, scalability, and ease of deployment while minimizing the risk of interference between components. The

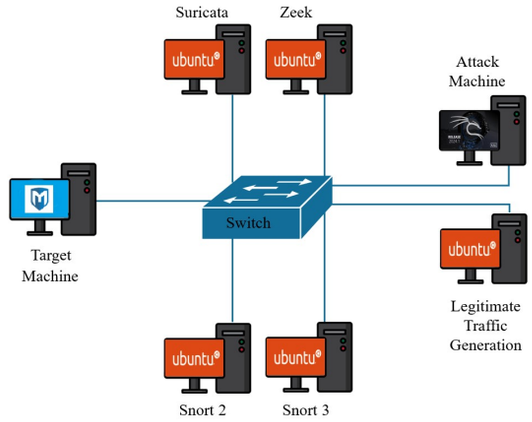



Fig. 1: Test-bed network

environment used was VMware Workstation 16 Pro running on a laptop

with an Intel® Core™ i7-8650U CPU, 32GB of RAM, and 1TB SSD storage, running Windows 10 Pro version 22H2. The testbed network, illustrated in Figure 1, consisted of seven virtual machines (VMs) connected to a virtual switch with a 1Gbps link to replicate real-world SME network speeds. Each VM was provisioned with adequate resources to ensure optimal performance during the experiments.

While manual downloading and installation of rules is feasible, using a management tool is advisable. PulledPork is a Perl-based utility that automates Snort rule management, facilitating the download, update, and maintenance of rule sets and IP block-list updates. It offers flexible policies (connectivity, balanced, security, or max-detect) and provides comprehensive feedback. PulledPork is compatible with Proofpoint ET and Cisco Snort rules, licensed under the GNU General Public License for commercial use, and can also be used with Suricata. With the release of Snort 3, PulledPork was redeveloped in Python 3 as PulledPork3. This version utilizes the LightSPD package and allows a single ruleset package to adjust its rules based on the engine version operating on the system, enabling users to choose a default policy for the ruleset. Suricata features a specialized tool for rule set management called suricata-update. This tool is the official method for updating and managing Suricata's rules and is included with Suricata starting from version 4.1. The Zeek Package Manager allows Zeek users to install third-party scripts and

plugins, functioning as a command-line script that requires Zeek to be installed locally. PulledPork and PulledPork 3 were selected for Snort 2 and Snort 3, respectively, to automate rule management due to their compatibility with Snort's rulesets. Suricata-update was utilized for Suricata, as it is the official tool for managing Suricata rulesets.

The selection of appropriate evaluation metrics is crucial for effectively assessing the strengths and weaknesses of the NUTs. The chosen metrics align with the research objectives of identifying the most suitable NUT for SMEs. The ease of deployment and management of each NUT will be assessed through the following criteria:

- Installation Complexity: Evaluation of how straightforward or challenging it is to install the IDS, including the number of steps, dependencies, and potential issues.
- Configuration Complexity: Assessment of the user interface and configuration options for intuitiveness and ease of use, covering aspects like rule management and alert generation.
- Management Overhead: Evaluation of the ongoing effort required to manage the NUT, considering the typical technical expertise available in SMEs.

The evaluation method includes installing and configuring each NUT on the virtual machine acting as the NIDS sensor, documenting the time taken for installation and configuration, and evaluating ongoing management tasks such as updates and rule management. The resource consumption of each NUT will be measured under simulated network traffic loads typical of SME activity, monitoring the following metrics:

- CPU Utilization: Percentage of CPU utilization by the NUT process, as high usage may indicate performance bottlenecks.
- Memory Usage: Amount of memory consumed by the NUT process; excessive usage could delay threat detection.
- Packet Drop Rate: Percentage of packets dropped by the NUT, where a higher rate indicates poorer performance.

For performance evaluation, resource consumption metrics (CPU and memory) will be monitored during network traffic simulations using the htop tool. Packet drop rates will be retrieved from the NUT outputs after each run, and custom bash scripts will be developed to parse and aggregate data from the logs. The collected data will be analyzed to determine the impact of traffic load on resource consumption and packet dropping for each NUT. The detection capability of each NUT is influenced by the availability of corresponding rules within the default rule set. This evaluation assesses how well the default rules cover a range of attacks by simulating various attack scenarios and analyzing the NUT's responses. The detection capabilities will be evaluated based on the following metrics:

- TPR: This measures how accurately each NUT detects real attacks, crucial for evaluating its efficiency in recognizing harmful activity.
- FNR: This metric addresses the failure to detect attacks, critical for preventing significant risks, especially for ransomware.

For detection capability evaluation, alerts generated during traffic replay will be collected and analyzed for each scenario, confirming their alignment with the identified attack type and documenting any missed detections. The performance evaluation experiments aim to assess the NUTs in terms of CPU utilization, memory consumption, and packet drop rate under various conditions. Key factors influencing NIDS performance include traffic rate, packet size, capturing methods, detection engine algorithms, ruleset size, and network flow types. The following scenarios were implemented for performance evaluation:

- Scenario 1: Baseline Performance Assessment establishes a baseline for employing NIDS within an SME network using a standard configuration and ruleset for each NUT. A TCP stream was generated using Iperf with a packet size of 1500 bytes, and performance was assessed as throughput increased from 100 Mbps to 1000 Mbps.
- Scenario 2: Ruleset Size Standardization addresses the varying sizes of the default rulesets across NUTs, implementing a standardized size of 10,000 rules for fair

comparison. Zeek was excluded due to its limited default ruleset.

- Scenario 3: Packet Capture Technique Evaluation explores the impact of different packet capture techniques (libpcap and afpacket) on performance, selecting the method with the least packet drop rate for further tests.
- Scenario 4: Detection Engine Optimization modifies the detection engine using the best-performing packet capture technique identified previously. The top three algorithms for each system were evaluated under the same network traffic conditions.
- Scenario 5: Impact of Ruleset Size on Performance examines how varying ruleset sizes (from 10,000 to 50,000 rules) affect NUT performance under different traffic loads (100 Mbps, 500 Mbps, and 1000 Mbps), excluding Zeek due to its ruleset limitations.
- Scenario 6: Real-World Traffic Simulation simulates daily activities using the BigFlows. pcap dataset, capturing typical TCP and UDP traffic. Traffic was redirected at 100 Mbps throughput to the NUTs.

For detection capability assessment, various scenarios were designed to evaluate the NUTs against common cyberattacks targeting SMEs. Each scenario focused on specific attack types:

- Scenario 1: DoS and Port Scanning Attacks involved simulating DoS attacks (SYN, ICMP, and UDP flooding) and a port scan using NMAP.
- Scenario 2: General Malware tested detection capabilities against a wide range of malware using 420 small pcap files containing malicious traffic.
- Scenario 3: Ransomware Attacks focused on detecting ransomware using 17 pcap files representing typical ransomware traffic.
- Scenario 4: Web Application Attacks evaluated responses to web-based attacks using OWASP ZAP and Metasploit, focusing on vulnerabilities such as SQL injection.
- Scenario 5: Phishing Attacks simulated phishing attempts, capturing traffic to assess the NUTs' ability to identify malicious URLs and domain spoofing.

Each scenario was executed for 180 seconds, with detection capability scenarios run until all related traffic was replayed. All experiments were repeated 10 times to ensure statistical significance, averaging results to mitigate the impact of outliers and provide a comprehensive understanding of the NUTs' performance under consistent conditions.

The Universe repository for Ubuntu 22.04 currently does not include the latest version of Snort 2. As shown in Figure 2, the repository's latest version is (2.9.15.1), whereas the Snort website offers version (2.9.20). Consequently, we will proceed to install the most recent version of Snort 2 directly from the source code.

The installation of Snort 2 follows these five steps:

*1) Update Ubuntu packages.*

```
$ sudo apt update && sudo apt dist-upgrade -y
```

*2) Install dependencies, Snort 2 has some prerequisites that need to be installed.*

```
$ sudo apt update && sudo apt dist-upgrade -y
,→ libdumbnet-dev build-essential flex
,→ bison zlib1g-dev libluajit-5.1-dev
,→ openssl libssl-dev liblzma-dev
,→ libnghttp2-dev
```

*3) Download some source tarballs and other files and store them in a folder for easy management.*

```
$ mkdir snort-src
$ cd snort-src/
```

*4) Install daq, Download and install the latest version of DAQ from the Snort website.*

```
$ wget https://www.snort.org/downloads/snort/
→ , daq-2.0.7.tar.gz
$ tar -zxvf daq-2.0.7.tar.gz
$ cd daq-2.0.7/
$./configure
$ make
```

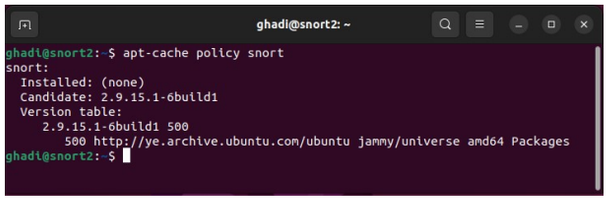



Fig. 2: Screenshot: latest Snort 2 release available in Ubuntu repository

```
$ sudo make install
$ cd ../
```

5) *Install Snort 2, Download and install the latest version of Snort 2 from the Snort website. The last version is 2.9.20, which released on 2022/05/13.*

```
$ wget https://www.snort.org/downloads/snort/
→ , snort-2.9.20.tar.gz
$ tar -xzvf snort-2.9.20.tar.gz
$ cd snort-2.9.20
$./configure --enable-sourcefire
$ make
$ sudo make install
```

Table I summarizes the key metrics observed during the implementation of each NUT, reflecting the practical challenges faced during deployment.

TABLE I
SUMMARY OF IMPLEMENTATION METRICS

| Criteria | Snort 2 | Snort 3 | Suricata | Zeek |
|---|---|---|---|---|
| Installation Time | mins 35 | mins 77 | mins 11 | mins 19 |
| Configuration Complexity | Moderate | Moderate | Easy | Difficult |
| Management Overhead | High | Moderate | Low | High |

Key observations indicate the following regarding the installation process:

- Snort 2: Installation took approximately 35 minutes due to manual compilation and dependency issues, which may be challenging for users with limited experience.
- Snort 3: Installation was longer at about 77 minutes, due to complex dependencies and configuration steps, making it less suitable for rapid deployment.
- Suricata: The fastest installation at only 11 minutes, benefiting from standard repositories and automated dependency management, ideal for SMEs needing quick setups.
- Zeek: Took around 19 minutes, relatively straightforward but required familiarity with network interface setup.

Regarding configuration complexity, the findings are as follows:

- Snort 2 and 3: Both versions rated as having moderate complexity. Snort 2 required manual edits, while Snort 3 needed knowledge of Lua-based adjustments.
- Suricata: Easiest to configure with a clear YAML-based setup, allowing simple adjustments of critical parameters.
- Zeek: Most complex due to its script-driven configuration, posing a significant barrier for SMEs without scripting knowledge.

In terms of management overhead:

- Snort 2: High management overhead due to frequent manual updates and maintenance of rulesets, which can be burdensome for SMEs.
- Snort 3: Reduced overhead compared to Snort 2 but still required moderate effort to maintain rulesets.
- Suricata: Lowest management overhead, with the suricataupdate tool automating rule updates, making it ideal for SMEs.
- Zeek: High management overhead due to the need for manual updates and maintenance of custom scripts, overwhelming for users lacking specialized skills.

The analysis indicates that Suricata is the most straightforward NUT to deploy and manage, making it an excellent choice for SMEs due to its ease of deployment and minimal management overhead. Specifically, Suricata's 11-minute installation and userfriendly configuration demonstrate its accessibility, while automated rule updates significantly reduce ongoing management efforts. Conversely, Snort and Zeek present higher barriers to entry due to complex configurations and management overhead. Snort demands more from

users, while Zeek's reliance on custom scripting limits accessibility without dedicated security staff.

## IV. Performance Evaluation

This section addresses the second research question (RQ2), examining the performance differences among open-source NIDS solutions Snort 2, Snort 3, Suricata, and Zeek, particularly in CPU usage, memory consumption, and packet dropping rates under diverse network conditions. Through a systematic evaluation of these metrics in various scenarios, the section delivers an in-depth analysis of each NIDS's resource efficiency and scalability, aiding in identifying the most appropriate solution for SMEs with limited resources.

### *A. Baseline Performance Assessment*

In this scenario, each NUT was evaluated using its default configuration and ruleset as network throughput increased from 100 Mbps to 1000 Mbps.

As shown in Figure 3, Snort 2 displayed a steady rise in CPU utilization, peaking at approximately 22.8% at 1000 Mbps. Its memory usage remained stable at 8.8%, but the packet drop rate rose significantly, from 0% at 100 Mbps to about 9.9% at 1000 Mbps. These results indicate that Snort 2 struggles under heavier loads, leading to packet drops and decreased detection capabilities. This issue stems largely from its single-threaded design, which restricts efficient packet processing.

In contrast, Snort 3 (Figure 3) recorded higher CPU utilization across all throughput levels, peaking at around 25.5% at 1000 Mbps. Although its memory usage was low at 1.76%, it also experienced packet drops—albeit less severe than Snort 2—starting at 0% and rising to 5.26% at maximum throughput. This indicates that while more memory-efficient, Snort 3 still relies heavily on CPU resources and has challenges with high traffic volume due to its default single-threaded processing. Nonetheless, it outperformed Snort 2 even in this constrained mode.

Suricata's multi-threaded architecture delivered a balanced performance, with CPU usage climbing to about 19% at 1000 Mbps while keeping memory

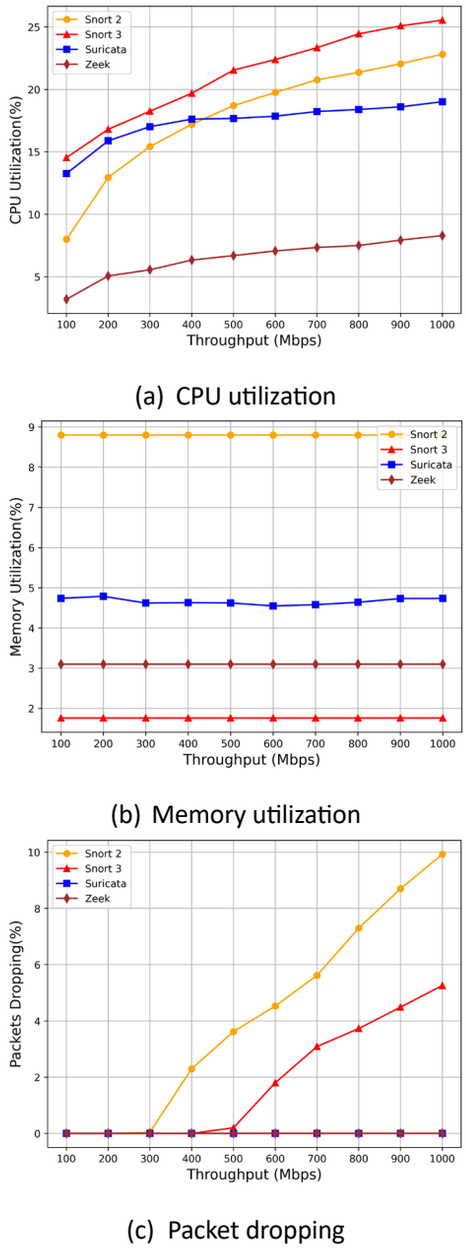



Fig. 3: Performance comparison of NIDS tools with default settings

usage low at approximately 4.7% (Figure 3). Remarkably, Suricata avoided any packet drops, even with the largest default ruleset among the NUTs, showcasing its capability to handle heavy traffic effectively.

Zeek excelled with extremely low CPU utilization, beginning at 3.2% and reaching only 8.3% at the highest throughput, alongside consistent memory

usage at 3.1%. Zeek maintained a 0% packet drop rate across all throughput levels, demonstrating its robustness without performance degradation (Figure 3). However, its lighter ruleset likely contributes to its lower resource consumption and negligible packet drops.

### *B. Ruleset Size Standardization*

To ensure a fair comparison, the rule set sizes for Snort 2, Snort 3, and Suricata were standardized to 10,000 rules each. As shown in Figure 4, Snort 2 experienced a slight decrease in CPU and memory usages compared to the baseline, with 22.0% at 1,000 Mbps, while its memory usage was constant at 8%. With the reduced rule set size, there was a slight improvement in the packet drop rate, yet it remained high at 9.1% at maximum throughput, indicating that rule set size impacts Snort 2's efficiency.

Snort 3 also exhibited reduced CPU and memory utilization compared to the baseline, reaching 25.4% at 1000 Mbps, while its memory decreased to 1.6%. The packet drop rate showed a modest improvement, with a final rate of 5.1% at maximum throughput. These results indicate that while Snort 3 benefits from ruleset standardization, it still struggles with high throughput scenarios, similar to Snort 2.

Suricata continued to perform well, with CPU utilization slightly lower than in the baseline, reaching 15.9% at 1000 Mbps. Memory usage was even more efficient at 2.05%, and it still recorded zero packet drops across all throughput levels. This reinforces Suricata's scalability and efficiency, making it particularly suitable for environments where high traffic and large rulesets are common.

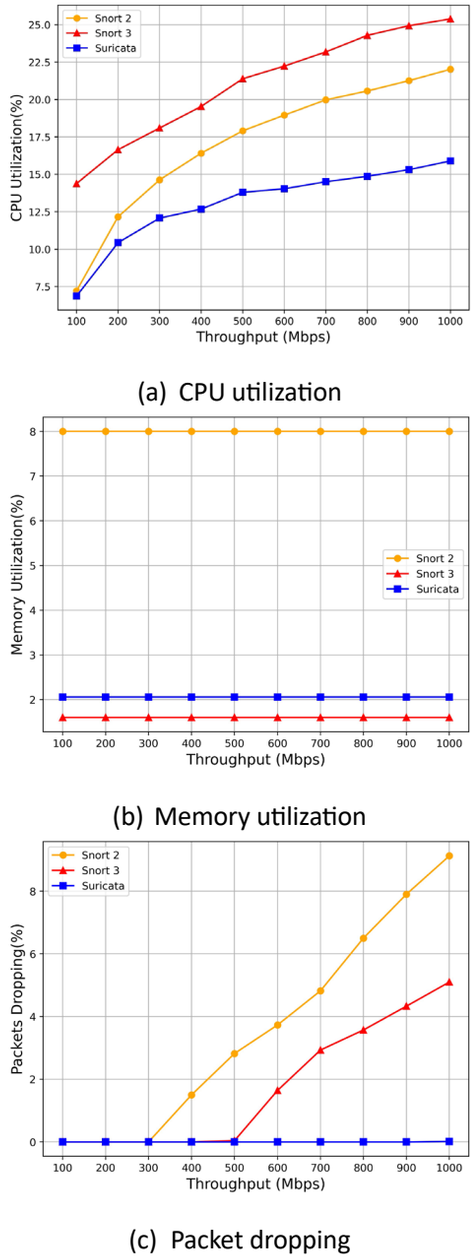
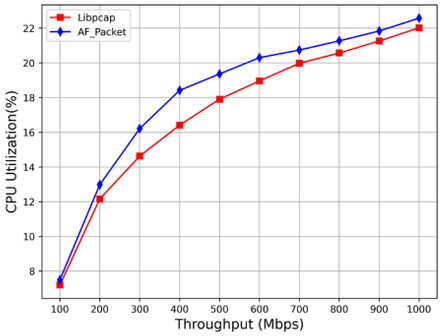



Fig. 4: Comparison of NIDS solutions with the same ruleset size

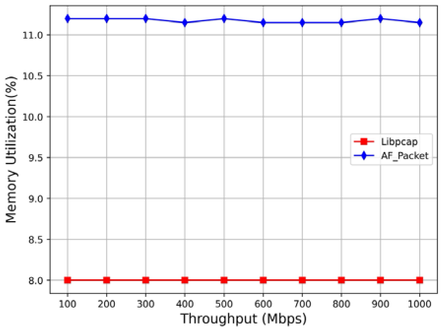
### *C. Packet Capture Technique Evaluation*

This scenario examined the effects of two prevalent packet capture methods, libpcap and afpacket, on the performance of each NUT.

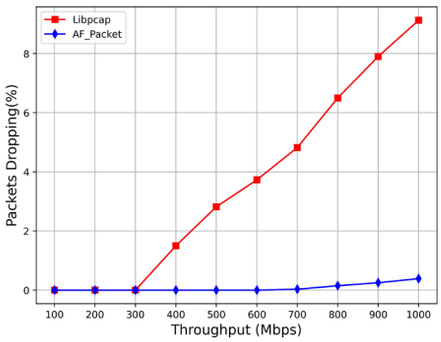
Figure 5 indicates that Snort 2 experienced an enhancement with the use of afpacket, as evidenced by a minor reduction in CPU utilization and a significant decrease in packet drop rates. At a network speed of 1000 Mbps, CPU usage was approximately 22.5%, and the packet drop rate was reduced to 0.39%, a substantial improvement from the 9.1% observed with libpcap. Although there was a slight increase in memory usage to 11.2%, packet handling was notably improved, with no drops occurring up to 700 Mbps. This demonstrates that despite afpacket's marginally higher memory requirement, the trade-off is justified by the gains in




(a) CPU utilization




(b) Memory utilization



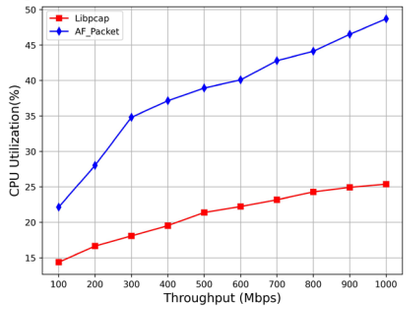

(c) Packet dropping

Fig. 5: Performance of Snort 2 with libpcap and afpacket

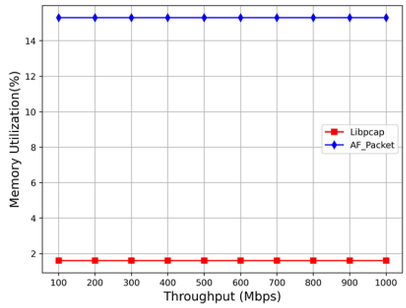
throughput and the reduction in packet loss.

Figure 6 demonstrates that Snort 3 has significantly advanced with afpacket, showing greater CPU utilization than libpcap, reaching 48.7% at 1000 Mbps and a rise in memory usage to 15.3%, while successfully reducing packet drops at almost all throughput levels. Due to libpcap missing proper load balancing for Snort 3 packet processing threads, it was not a valid choice for Snort 3 multi-threading.

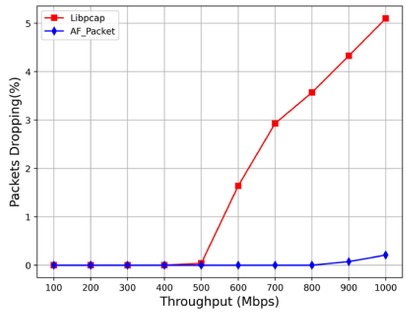
nfigurable detection engine, and therefore, nfiguration.




(a) CPU utilization




(b) Memory utilization




(c) Packet dropping

Fig. 6: Performance of Snort 3 with libpcap and afpacket

Snort 3 utilizes afpacket for load balancing network traffic between each packet processing thread, which was the only option for Snort 3 multi-threading when using Snort 3 to analyze realtime network traffic. It can be configured in the Snort 3 config file or command line, as seen in Section ??.

This balance between heightened CPU and memory usage against decreased packet loss suggests that afpacket enhances Snort 3's detection abilities notwithstanding the increased consumption of resources.

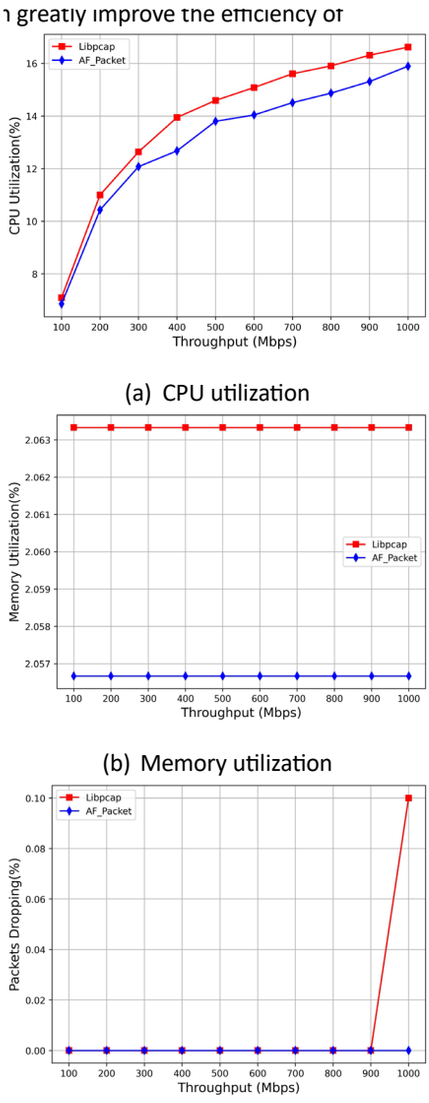



Fig. 7: Performance of Suricata with libpcap and afpacket

both capture techniques, showing very low CPU utilization (15.9% with afpacket), maintaining its low memory usage (around 2%), and zero packet drops throughout. This indicates that Suricata's efficiency is less dependent on the packet capture method, underscoring its overall robustness.

Figure 8 shows, that Zeek sustained outstanding performance with afpacket, demonstrating very low CPU and memory usage, along with zero packet drops. The negligible impact of the capture method on Zeek's performance underscores its

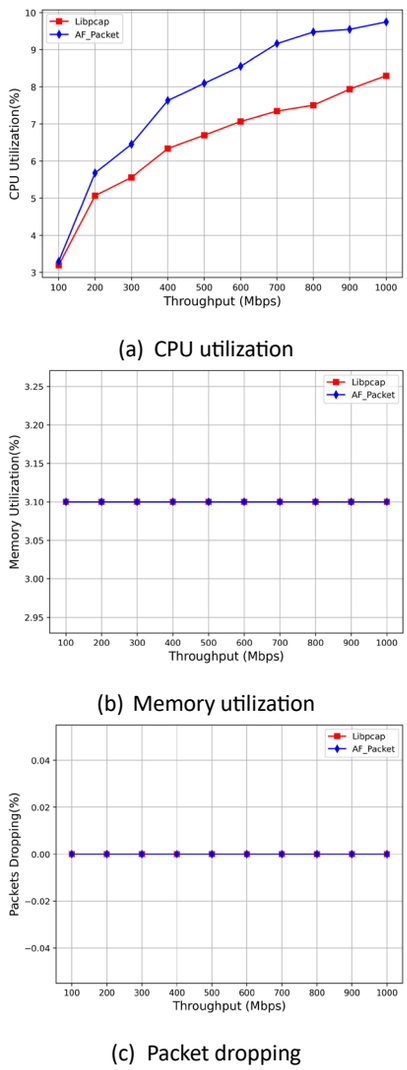



Fig. 8: Performance of Zeek with libpcap and afpacket

appropriateness for settings where stable and reliable detection is paramount.

The comparative performance of all NUTs with afpacket is further highlighted in Figure 9, which emphasizes the importance of selecting optimal packet capture methods. For SMEs, implementing afpacket with their chosen NIDS could be a straightforward adjustment to improve overall network security performance.

### *D. Detection Engine Optimization*

The detection engine optimization experiments explore the impact of different detection engine

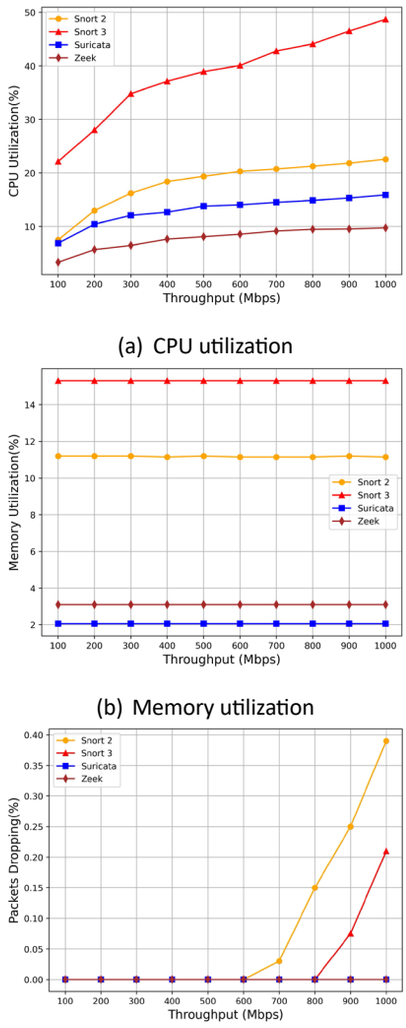



Fig. 9: Performance comparison of NIDS tools with afpacket

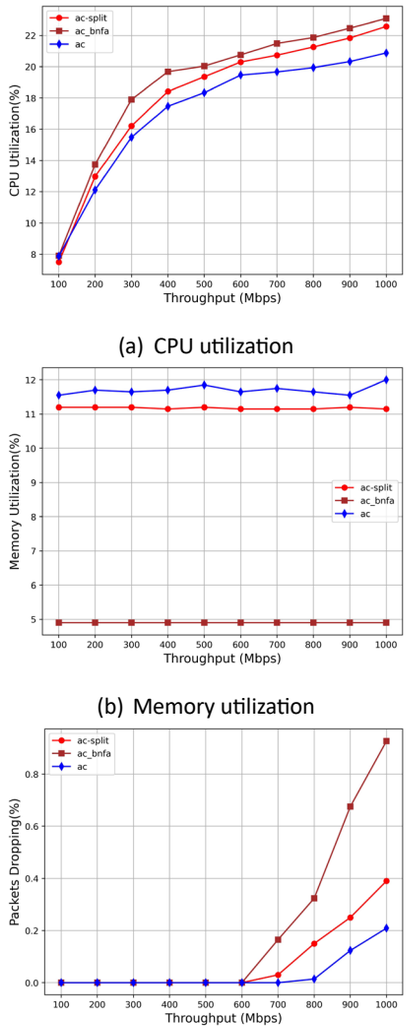
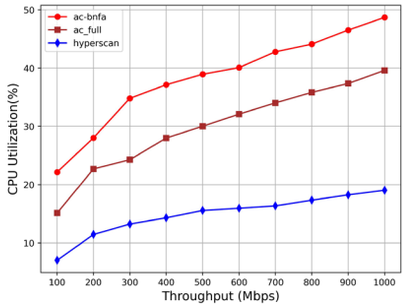



Fig. 10: Performance of Snort 2 with different detection engine

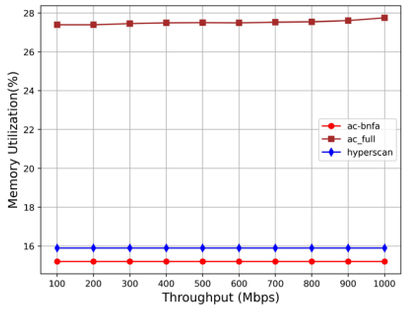
configurations on the performance of Snort 2, Snort 3, and Suricata, each using afpacket as the efficient packet capture method determined in the preceding scenario. It is important to note that Zeek was not included in this assessment due to its non-configurable detection engine, and therefore, it was kept in its default configuration.

While the ac option in Snort 2 did lead to a modest reduction in CPU usage, down to 20.0% at 1000 Mbps, and a decrease in packet drop rate to 0.21%, indicating improved efficiency and packet processing capacity, it also caused a significant increase in memory usage as shown in Figure 10. Our testings showed that Snort configured with the ac option experienced substantial memory consumption, which worsened with larger rule sets. An attempt to run Snort with a 50,000 rule set failed due to memory limitations. Resulting in, switching to the alternative ac-split option.

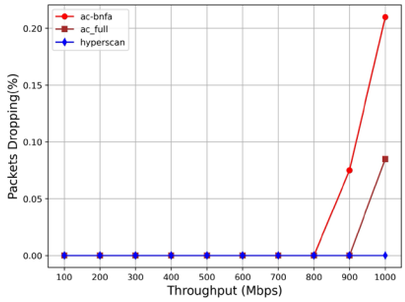
As shown in Figure 11, Snort 3's hyperscan configuration provided substantial performance benefits, with CPU utilization greatly reduced and memory usage stabilized at 15.9%. This configuration enabled Snort 3 to handle high throughput without packet loss, making it more competitive with Suricata's consistently low memory and CPU usage.




(a) CPU utilization

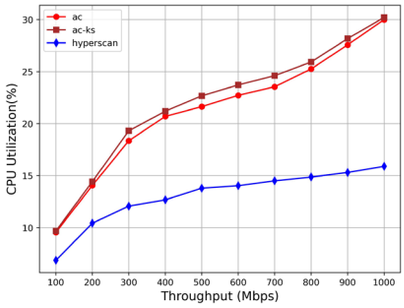



(b) Memory utilization

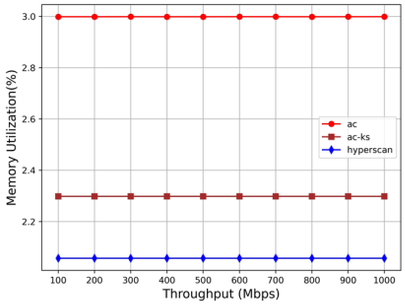



(c) Packet dropping

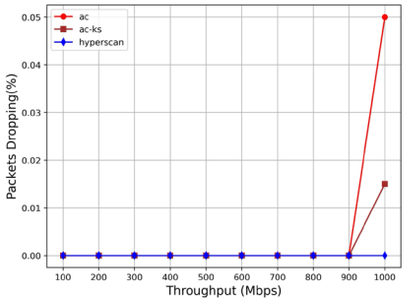
Fig. 11: Performance of Snort 3 with different detection engine




(a) CPU utilization




(b) Memory utilization




(c) Packet dropping

Fig. 12: Performance of Suricata with different detection engine

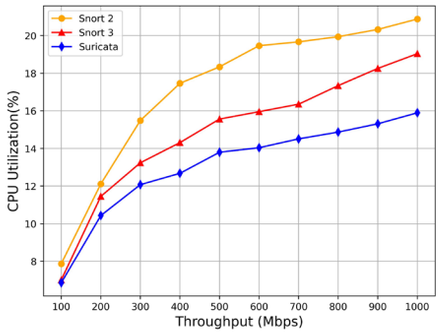
Figure 12 illustrates that the default Suricata configuration, utilizing afpacket with hyperscan, maintained optimal performance, further confirming the efficiency of the default Suricata setup as previously discussed.

The comparative performance of all NUTs, paired with the optimal detection engine for each, is further illustrated in Figure 13. This underscores the criticality of choosing the optimal detection engine. For SMEs, the adoption of advanced detection engines such as hyperscan can greatly improve the efficiency of NIDS, particularly in scenarios with high traffic volumes and sophisticated attack vectors.

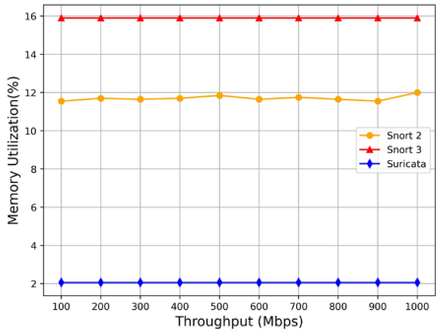
### *E. Impact of Ruleset Size*

The results of this scenario highlight the significant impact of ruleset size on the performance of Snort 2, Snort 3, and Suricata under various traffic conditions, including low, moderate, and peak throughput. As the ruleset size increased from 10,000 to 50,000 rules, all NIDS solutions showed a rise in CPU utilization and memory consumption, with Snort 2 experiencing the most pronounced increase in packet dropping rates, particularly at higher traffic loads.

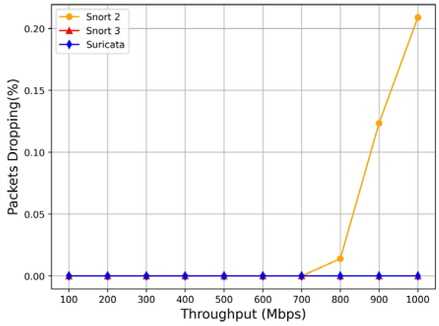
Under low throughput (100 Mbps), Figure 14, Snort 2's CPU usage gradually rose from 7.49% with 10,000 rules to 10.33% with 50,000 rules, while its




(a) CPU utilization




(b) Memory utilization




(c) Packet dropping

Fig. 13: Performance comparison of NIDS tools with optimized configuration

memory usage increased significantly, indicating a more resource-intensive operation as ruleset sizes expanded. Notably, Snort 2 maintained zero packet drops under low traffic conditions but exhibited a rising trend in memory usage that could impact performance at higher loads. Snort 3 demonstrated similar trends in CPU and memory usage, though it managed to maintain no packet drops across all ruleset sizes under low throughput, indicating a better efficiency in handling increasing rulesets compared to Snort 2. Suricata exhibited the lowest

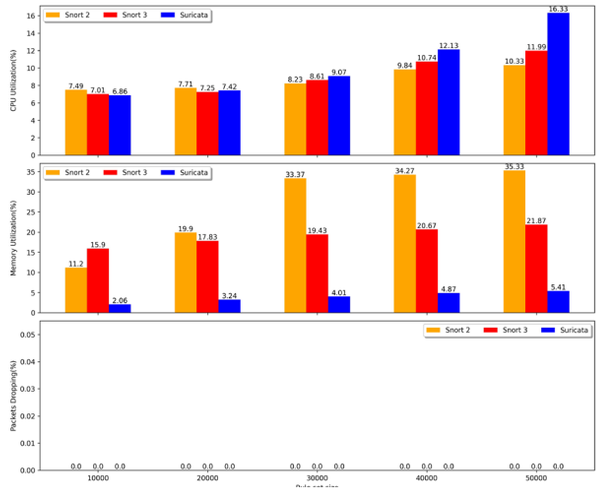



Fig. 14: Performance comparison of NIDS tools against ruleset size under low throughput (100 Mbps)

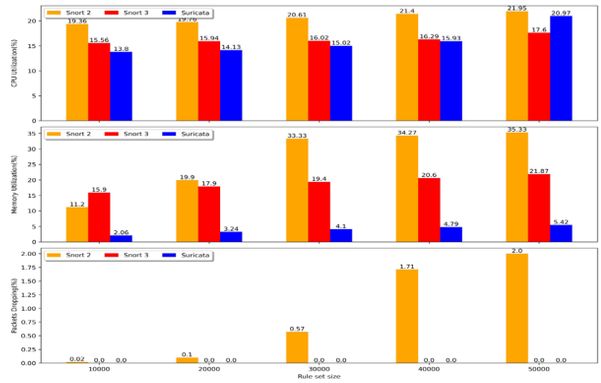



Fig. 15: Performance comparison of NIDS tools against ruleset size under moderate throughput (500 Mbps)

CPU and memory usage under the same conditions, maintaining zero packet drops, suggesting a high level of efficiency even with larger rulesets.

At moderate throughput (500 Mbps), Figure 15, the performance gaps widened. Snort 2 began showing packet drops starting from the 20,000-rule size, escalating to 2% at 50,000 rules, which is concerning for environments requiring high reliability. The increase in packet drop rates suggests that Snort 2 struggles with scalability under increased traffic loads and larger rulesets, making it less suitable for high-performance requirements without significant tuning. Snort 3 managed to avoid packet drops until the largest ruleset size, demonstrating better adaptability and scalability under moderate conditions. Suricata continued to show the most efficient performance,

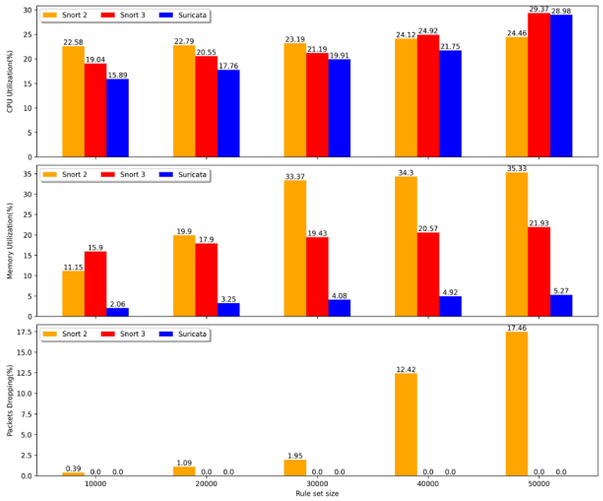



Fig. 16: Performance comparison of NIDS tools against ruleset size under peak throughput (1000 Mbps)

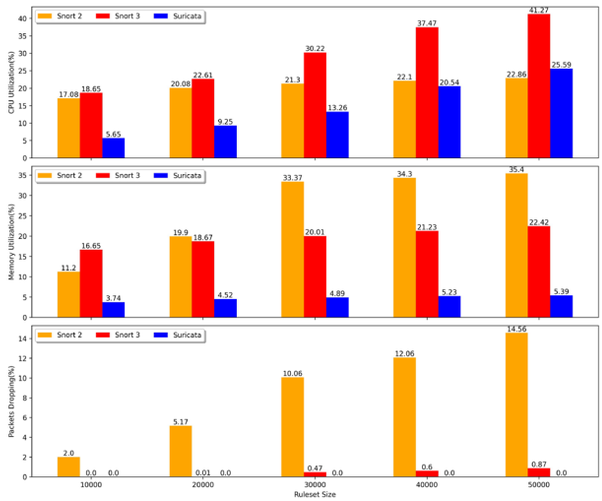



Fig. 17: Performance comparison of NIDS solutions against the BigFlow.pcap dataset with different ruleset size

maintaining zero packet drops and relatively low CPU and memory usage, further confirming its capability to handle large rule sets and moderate traffic without performance degradation.

Under peak throughput (1000 Mbps), Figure 16, the differences became starkly apparent. Snort 2's packet drop rate surged dramatically to 17.46% at the largest ruleset size, highlighting severe limitations in high-traffic scenarios. This performance suggests that Snort 2's detection engine struggles under heavy loads, particularly when coupled with extensive rulesets, potentially compromising security in high-traffic environments. Snort 3 and Suricata demonstrated much better resilience, with Suricata maintaining zero packet drops and only moderate increases in resource consumption. This performance underscores Suricata's robust architecture, making it particularly suitable for highthroughput environments typical of larger SMEs or those with intensive security monitoring requirements.

### *F. Performance Comparison with Real-World Traffic*

This scenario evaluated the performance of Snort 2, Snort 3, and Suricata using the BigFlows.pcap dataset, which replicates realistic network conditions with diverse traffic types, including HTTP browsing, file transfers, and chat applications. The dataset featured over 40,686 distinct flows with a range of packet sizes from 60 to 1514 bytes, and an average packet length of 449 bytes. This scenario allowed for a comprehensive assessment of how each NIDS handles complex, mixed traffic patterns typical of SME environments. Key factors affecting performance were the number and variety of flows and the variability in packet size, both of which significantly influence resource consumption, and packet drop rates.

The combination of numerous, varied flows and diverse packet sizes directly challenged the detection engines of Snort 2, Snort 3, and Suricata, highlighting their strengths and weaknesses in handling real-world traffic dynamics.

Figure 17 illustrates that Snort 2 had difficulties handling the growing number and diversity of flows, along with the variability in packet sizes. At a ruleset size of 10,000, Snort 2 maintained acceptable performance with a CPU utilization of 17.08% and memory usage at 11.2%, alongside a packet drop rate of 2%. However, as the ruleset size grew to 50,000, CPU utilization increased to 22.86%, and memory consumption surged to 35.4%. More critically, the packet dropping rate escalated sharply to 14.56%, indicating a significant decline in Snort 2's ability to process traffic effectively. This steep increase in packet drops suggests that Snort 2 may struggle with the dynamic nature of actual network traffic, where multiple concurrent flows and varying packet sizes can challenge its detection capabilities. The pronounced performance

degradation with larger rulesets poses a risk of undetected threats, making Snort2 less suitable for SMEs that require reliable threat monitoring with minimal packet loss.

Snort 3 Performance Analysis: Snort 3 showed a more robust performance profile compared to Snort 2, although it still faced challenges at higher ruleset sizes. At 10,000 rules, Snort 3 exhibited a CPU utilization of 18.65% and memory usage of 16.65%, with no packet drops, indicating efficient handling of traffic with relatively light computational demands. As the ruleset expanded to 50,000, CPU utilization increased significantly to 41.27%, and memory usage reached 22.42%. Despite this increase in resource consumption, Snort 3 managed to keep packet drops to a minimum, only recording a drop rate of 0.87% at the highest ruleset size. This low packet drops rate highlights Snort 3's enhanced detection engine and better resource optimization, which allows it to handle complex traffic with higher detection accuracy and minimal loss. However, the elevated CPU usage underlines potential performance bottlenecks, suggesting that while Snort 3 can manage large rulesets effectively, it does so at the cost of significantly increased processing power, which could impact overall system performance if not adequately provisioned.

Suricata Performance Analysis: Suricata consistently outperformed both versions of Snort across all metrics, maintaining superior detection capabilities with minimal resource consumption and packet drops. At a ruleset size of 10,000, Suricata's CPU utilization was remarkably low at 5.65%, and memory usage was 3.74%, with zero packet drops, showcasing its highly efficient architecture. As the ruleset size increased to 50,000, Suricata's CPU usage rose to 25.59%, and memory consumption increased moderately to 5.39%. Despite the larger ruleset, Suricata maintained a perfect record with zero packet drops, demonstrating its ability to efficiently scale and process high volumes of traffic without compromising performance. This robust performance can be attributed to Suricata's multi-threaded processing and optimized detection algorithms, which allow it to balance load effectively across system resources, making it particularly wellsuited for real-world deployments in SMEs where reliability and efficiency are paramount.

## V. Detection Capability Assessment

This section explores the third research question (RQ3), focusing on the efficacy of Snort 2, Snort 3, Suricata, and Zeek in detecting common cyberattacks that frequently target SMEs. It assesses each NIDS detection accuracy through controlled attack scenarios, such as DoS, malware, ransomware, web application attacks, and phishing. The evaluation measures the TPR and FNR of each NIDS. The findings provide insights into the strengths and weaknesses of each tool's ability to protect SME networks from diverse threats.

### *A. Scenario 1: DoS and Port Scanning Attacks*

The results of this scenario show significant differences in how each NUT handles DoS and port scanning attacks. Suricata outperformed the other systems, achieving an impressive TPR of 85% with a relatively low FNR of 15%. This reflects Suricata's superior ability to manage high-throughput traffic and accurately detect patterns associated with DoS and scanning activities. Snort 3 followed with a TPR of 80%, showing improvements over Snort 2, which only managed a TPR of 75%. Zeek, on the other hand, displayed a lower TPR of 60%, with a high FNR of 40%, which can be attributed to its focus on behavioral analysis rather than signature-based detection. Zeek's relative under-performance in this scenario indicates its limitations in detecting rapid, highvolume attacks like DoS, where signature-based systems like Suricata excel.

### *B. Scenario 2: General Malware*

In the general malware detection scenario, Suricata once again demonstrated its strength, achieving the highest TPR of 90% with only a 10% FNR. This result highlights the effectiveness of Suricata's robust signature-based detection, which allows it to identify a wide range of malware threats. Snort 3 performed well with a TPR of 86%, significantly outperforming Snort 2, which had a TPR of 77%. The improved detection rate in Snort 3 can be attributed to enhancements in its rule set and detection engine. Zeek, with its behavior-based approach, achieved a TPR of 78%, which is slightly higher than Snort 2 but lower than Snort

3 and Suricata. Zeek's detection capabilities in this scenario were hampered by its reliance on behavioral indicators rather than signature matching, which limited its ability to identify malware without clear behavioral anomalies.

### *C. Scenario 3: Ransomware Attacks*

The ransomware attack scenario underscores Suricata's capability to detect sophisticated threats, with a TPR of 94% and the lowest FNR at 6%. This demonstrates Suricata's effectiveness in recognizing ransomware-specific traffic patterns, including command and control, data exfiltration, and encryption activities. Snort 3 also performed well with a TPR of 87%, while Snort 2 lagged slightly behind at 81%. Zeek's performance was notable, with a TPR of 88%, indicating that its behavioral detection approach is particularly useful in identifying ransomware traffic, which often exhibits distinctive behaviors. However, Zeek's slightly higher FNR (12%) compared to Suricata suggests that it might miss more subtle or encrypted ransomware activities that rely less on obvious behavioral signatures.

### *D. Scenario 4: Web Application Attacks*

Web application attacks, such as SQL Injection and CrossSite Scripting, were well detected by most NUTs, but Suricata continued to lead with a TPR of 91% and an FNR of 9%. Snort 3 followed closely with a TPR of 88%, highlighting its competence in handling web-based threats. Snort 2 trailed with a TPR of 85%, indicating that while it is effective, it is not as finely tuned as its successor or Suricata. Zeek's performance in this scenario was strong, with a TPR of 89%, reflecting its proficiency in monitoring HTTP traffic and identifying anomalous behaviors associated with web application exploits. However, the slight variation in FNR between Zeek (11%) and Suricata (9%) indicates that signature-based systems may still have a slight edge in detecting web vulnerabilities.

### *E. Scenario 5: Phishing Attacks*

In the phishing scenario, Suricata demonstrated exceptional accuracy with the highest TPR of 95% and the lowest FNR of 5%, confirming its effectiveness in detecting malicious URLs, domain spoofing, and suspicious email traffic. Snort 3 performed similarly well, achieving a TPR of 90% and an FNR of 10%, outperforming Snort 2 (TPR 83%, FNR 17%). This indicates that while both versions of Snort are effective in detecting phishing attempts, Snort 3 offers better accuracy. Zeek also achieved strong results with a TPR of 89% and an FNR of 11%, showing its capability in identifying phishing-related behaviors such as credential harvesting and anomalous domain access. However, Suricata's higher TPR and lower FNR make it the most reliable NUT in detecting phishing threats, likely due to its extensive rule set coverage for domain-related attacks.

## VI. Conclusion, Recommendations, and Future Works

This study aimed to evaluate the effectiveness of three opensource network intrusion detection systems (NIDS): Snort 2, Snort 3, Suricata, and Zeek, in the context of small and mediumsized enterprises (SMEs). The research focused on ease of deployment, performance under varying network conditions, and detection accuracy for common SME cyber threats. Through controlled experiments in a virtualized environment simulating realistic SME conditions, we assessed the performance of these NIDS solutions.

The findings revealed that Suricata consistently outperformed the others in scalability, efficiency, and low packet drop rates, making it highly suitable for SMEs. Snort 3, when optimized with afpacket and hyperscan, demonstrated significant potential but is best suited for resource-rich environments. Snort 2 exhibited limitations under heavy traffic, while Zeek, though efficient, may not address all security needs due to its lighter ruleset.

These findings have important implications for SME NIDS deployment, highlighting the strengths of Suricata as a scalable solution. Snort 3 can benefit SMEs with sufficient hardware, while Snort 2 requires careful consideration in high-traffic scenarios. Future research should validate these findings in real-world environments to better understand NIDS performance dynamics.

This study offers valuable insights for SMEs aiming to enhance their cybersecurity posture. By clarifying the strengths and limitations of these NIDS, we contribute to improving network security in small and medium-sized enterprises.

Based on the study's findings, the following recommendations are proposed:

1. Prioritize Suricata for Performance and Scalability: SMEs should consider deploying Suricata due to its consistent performance and efficiency.
2. Deploy Snort 3 in Resource-Rich Settings: Snort 3 is best for powerful hardware environments, ensuring high detection accuracy.
3. Use Snort 2 for Low-Traffic Environments: This NIDS is Suited for simpler networks, but caution is advised in more Demanding scenarios.
4. Leverage Zeek for Light weight Monitoring: Zeek is ideal For basic visibility and monitoring in environments where Extensive detection is less critical.
5. Regularly Update and Optimize NIDS: SMEs should keep Their NIDS updated and optimized for performance.
6. Consider Hybrid Approaches: Combining different NIDS Can enhance overall security coverage.
7. Invest in Staff Training: Training IT staff is crucial for effective NIDS deployment and management.

Future research could explore integrating machine learning with NIDS to enhance detection capabilities, customize rule sets for specific industries, and develop user-friendly interfaces for non-technical users. Additionally, longitudinal studies examining the long-term impact of NIDS deployment in SMEs would provide valuable insights into their adaptability and sustainability over time.

## Funding

This article did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

## Conflict of Interest

Authors declare that they have no conflict of interest.

## References

[1] W. Park and S. Ahn, "Performance comparison and detection analysis in snort and suricata environment," *Wireless Personal Communications*, vol. 94, pp. 241–252, 2017.

[2] N. Rawindaran, A. Jayal, E. Prakash, and C. Hewage, "Perspective of small and medium enterprise (sme's) and their relationship with government in overcoming cybersecurity challenges and barriers in wales," *International Journal of Information Management Data Insights*, vol. 3, no. 2, p. 100191, 2023.

[3] F. A. Logic, "What is a network ids and why do you need it?" 10 2020. [Online]. Available: https://www.alertlogic.com/blog/what-isa-network-ids-and-why-do-you-need-it/

[4] E. Tsukerman, "What is an intrusion detection system (ids)," *Designing a Machine Learning Intrusion Detection System*, 2020.

[5] A. Chidukwani, S. Zander, and P. Koutsakis, "A survey on the cyber security of small-to-medium businesses: Challenges, research focus and recommendations," *IEEE Access*, vol. 10, p. 85701–85719, 2022.

[6] Y. Tayyebi and D. Bhilare, "A comparative study of open source network based intrusion detection systems," *Int. J. Comput. Sci. Inf. Technol. IJCSIT*, vol. 9, no. 2, pp. 23–26, 2018.

[7] C. Hoover, "Comparative study of snort 3 and suricata intrusion detection systems," Master's thesis, Undergraduate Honors Theses, Computer Science and Computer Engineering, 2022.

[8] A. Waleed, A. F. Jamali, and A. Masood, "Which open-source ids? snort, suricata or zeek," *Computer Networks*, vol. 213, p. 109116, 2022.

[9] S. A. R. Shah and B. Issac, "Performance comparison of intrusion detection systems and application of machine learning to snort system," *Future Generation Computer Systems*, vol. 80, pp. 157–170, 2018.

[10] 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.

[11] S. B. Chalmers, "Comparison of different security tools to detect risks in networks," *International Journal Of Computer Sciences and Mathematics Engineering*, vol. 1, no. 1, pp. 13–19, 2022.



[12] A. Alhomoud, R. Munir, J. P. Disso, I. Awan, and A. Al-Dhelaan, "Performance evaluation study of intrusion detection systems," *Procedia Computer Science*, vol. 5, pp. 173–180, 2011.

[13] D. Day and B. Burns, "A performance analysis of snort and suricata network intrusion detection and prevention engines," in *Fifth International Conference on Digital Society*, Gosier, Guadeloupe, 2011, pp. 187–192.

[14] E. Albin and N. C. Rowe, "A realistic experimental comparison of the suricata and snort intrusion-detection systems," in 2012 *26th International Conference on Advanced Information Networking and Applications Workshops*, 2012, pp. 122–127.

[15] R. Padmashani, S. Sathyadevan, and D. Dath, "Bsnort IPS better snort intrusion detection/prevention system," in *2012 12th International Conference on Intelligent Systems Design and Applications (ISDA)*. IEEE, 2012, pp. 46–51.

[16] W. Bulajoul, A. James, and M. Pannu, "Network intrusion detection systems in high-speed traffic in computer networks," in *2013 IEEE 10th International Conference on E-Business Engineering*, 2013, pp. 168–175.

[17] J. S. White, T. Fitzsimmons, and J. N. Matthews, "Quantitative analysis of intrusion detection systems: Snort and suricata," in *Cyber Sensing 2013*, vol. 8757. International Society for Optics and Photonics, 2013, p. 875704.

[18] M. Saber, M. G. Belkasmi, S. Chadli, M. Emharraf, and I. El Farissi, "Implementation and performance evaluation of intrusion detection systems under high-speed networks," in *Proceedings of the 2nd International Conference on Big Data, Cloud and Applications*, 2017, pp. 1–6.

[19] 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.

[20] Q. Hu, M. R. Asghar, and N. Brownlee, "Evaluating network intrusion detection systems for high-speed networks," in *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*. IEEE, 2017, pp. 1–6.

[21] Q. Hu, S.-Y. Yu, and M. R. Asghar, "Analysing performance issues of open-source intrusion detection systems in high-speed networks," *Journal of Information Security and Applications*, vol. 51, p. 102426, 2020.

[22] 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.

[23] 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.