Siber Güvenlik ve Dijital Ekonomi Dergisi

Cybersecurity and Digital Economy Journal

e - ISSN: 3108-3706
https://dergipark.org.tr/tr/pub/cdej

Publisher: Düzce University

Research Article

Design and Implementation of a Network Intrusion Detection System Using Snort 3 in a Virtualized Linux Environment

*Aly Abdelhalima,[a]**

[a] Oula, Al Giza, Giza Governorate, Egypt.
* **Corresponding author:** aliabdelhalim272@gmail.com

Article Information:
Received: 11/10/2025, Accepted: 17/11/2025

ABSTRACT

Network Intrusion Detection Systems (NIDS) are indispensable elements in modern cybersecurity infrastructures, tasked with identifying malicious traffic and preventing compromise in increasingly complex network environments. Open-source frameworks remain particularly valuable due to their transparency, adaptability, and widespread adoption in both academia and industry. Among them, Snort 3 represents a significant re-engineering of the widely deployed Snort 2, introducing Lua-based configurations, modular detection engines, and enhanced flow inspection capabilities. This study presents a practical, reproducible implementation of Snort 3 in a controlled virtualized testbed using UTM on Apple M1 hardware. The experimental design includes the simulation of three canonical cyberattacks: ICMP Ping reconnaissance, TCP Port Scanning with Nmap, and SYN Flood Denial-of-Service using hping3. Custom detection rules were authored and validated to measure Snort 3's responsiveness and accuracy under realistic traffic conditions. Results demonstrated 100% detection accuracy across all scenarios when configurations and rules were correctly implemented, aligning with findings in recent performance comparisons of Snort 3 and Suricata. However, challenges such as the strict syntax of Snort 3's Lua configuration and its sensitivity to misconfigured rule definitions underscore the potential for operational blind spots if tuning is neglected. Performance testing revealed that while Snort 3 is effective in detecting volumetric SYN flood traffic, single-threaded execution on constrained environments may limit scalability — a result consistent with contemporary benchmarks. Overall, this research contributes a hands-on reproducible framework for deploying Snort 3 in educational and small-scale production networks, emphasizing the critical role of precise configuration, continuous rule refinement, and performance monitoring in modern intrusion detection.

***Keywords*:** *Intrusion Detection System (IDS), Metasploitable, Cyberattack Simulation, Network Security, Open-Source Security, DDoS Detection*

I. INTRODUCTION

As cyber threats continue to grow in both volume and sophistication, organizations require increasingly advanced security mechanisms capable of identifying, analyzing, and mitigating malicious activities before critical assets are compromised. Recent reports highlight a sharp increase in cyberattacks leveraging multi-stage exploitation and stealthy reconnaissance techniques, often culminating in large-scale distributed denial-of-service (DDoS) campaigns or ransomware deployments. Traditional perimeter defenses such as firewalls, intrusion prevention systems, or antivirus tools are no

Cite as: Abdelhalima, A. (2025). Design and implementation of a network intrusion detection system using snort 3 in a virtualized linux environment. *Cybersecurity and Digital Economy Journal*, *1*(2), 111-119.

longer sufficient to handle these evolving threats, as they often rely on static filtering and are not designed to recognize the nuanced patterns of complex intrusions (Javaheri et al., 2023).

In this context, Network Intrusion Detection Systems (NIDS) play a pivotal role as a critical second line of defense, offering automated, real-time monitoring of network traffic and enabling timely responses to suspicious or malicious behaviors (Hnamte et al., 2024). Unlike firewalls, which enforce predefined access rules, or endpoint security systems, which focus on host-level defense, NIDS provide holistic visibility across entire network segments. They monitor traffic for known attack signatures, protocol anomalies, and behavioral deviations, thereby serving as an indispensable component of modern defense-in-depth strategies (Falowo et al., 2024).

A. Evolution of Open-Source IDS Platforms

Among open-source NIDS solutions, Snort has remained the industry benchmark for more than two decades due to its active rule community, modular architecture, and proven operational reliability. First released in 1998, Snort quickly became the most widely deployed open-source IDS, benefiting from an active ecosystem that produced thousands of rules for detecting malware, reconnaissance activity, and common exploitation techniques. Its adoption in both enterprise and academic contexts has made it a reference point in intrusion detection research, often used as a baseline for evaluating newer systems (Shah & Issac, 2017).

Competing open-source solutions have since emerged, most notably Suricata and Zeek (formerly Bro). Suricata's architectural advantage lies in its native multi-threading support, enabling it to handle high-throughput environments more gracefully than legacy Snort 2.x. Zeek, on the other hand, positions itself as a network monitoring and analysis framework, offering deep protocol parsing and event-driven scripting but lacking the same level of community-maintained signature sets as Snort and Suricata. Collectively, these tools illustrate the diversity of approaches in open-source IDS development (LevelBlue, 2024).

B. The Emergence of Snort 3

The release of Snort 3 (Snort++) marked a fundamental re-architecture of the platform, addressing many of the scalability and maintainability concerns associated with Snort 2.x. (Cisco, 2021; Snort Project, 2021) Key innovations include:

- Lua-based configuration: replacing the monolithic .conf format with a flexible scripting engine, allowing fine-grained customization.
- Modular detection engines: enabling dynamic loading of inspectors for different protocols (e.g., HTTP/2, TLS 1.3, IPv6).
- Improved flow handling: enhancing state tracking and scalability for modern, high-speed networks.
- Extensibility: providing hooks for machine learning integration, notably through SnortML introduced by Cisco Talos in 2024.

These innovations align Snort 3 with contemporary network demands, including encrypted traffic analysis, protocol heterogeneity, and cloud-native environments. However, the transition also introduces significant complexity. Snort 3's strict configuration syntax and sensitivity to rule formatting pose a steep learning curve, particularly for students and early-career practitioners (Dependency Hell, 2021).

C. Current Research Landscape

Recent comparative evaluations of Snort 3 and competing platforms such as Suricata suggest a nuanced performance profile. These studies demonstrate that while Snort 3 achieves competitive detection accuracy—often matching or exceeding Suricata in controlled lab environments—its performance under heavy traffic loads is constrained by its single-threaded execution model (Kumar & Singh, 2023; Ghazi et al., 2024). By contrast, Suricata scales more efficiently in environments exceeding 1 Gbps throughput, owing to its multi-core design. This dichotomy reinforces the notion that Snort 3 is well suited for small to medium-scale deployments, research laboratories, and educational environments, whereas Suricata may be preferred in ISP backbones or enterprise data centers (Tolumichael, 2025).

Another research direction involves the integration of machine learning (ML) with IDS platforms. Cisco Talos's release of SnortML represents a step toward hybrid detection systems capable of addressing polymorphic malware and zero-day exploits (Jaw et al., 2022).Independent academic contributions have also explored ML-assisted Snort rule generation, semi-automated anomaly detection, and hybrid IDS architectures combining signature-based and statistical methods .While promising, these approaches remain experimental and often lack reproducible testbeds for validation.

D. Gaps in the Literature

Despite significant advances, several gaps remain in the IDS literature:

1.Lack of reproducible, step-by-step Snort 3 deployment guides, especially tailored for non-traditional hardware platforms such as Apple Silicon (M1/M2).

2.Limited empirical validation of Snort 3 in educational labs compared to legacy Snort 2.x studies.

3.Insufficient focus on configuration errors as a source of detection failure—most existing studies assume flawless deployment, which is rarely the case in practice.

4.Scalability constraints in virtualized or resource-constrained environments, which remain underexplored despite being highly relevant for universities and small organizations.

E. Contributions of This Study

This paper addresses these gaps through a hands-on experimental deployment of Snort 3 in a controlled, virtualized Linux environment on Apple M1 hardware. The contributions of this study include:

•A step-by-step deployment guide for Snort 3 on Kali Linux using UTM virtualization.

•Development of custom detection rules for ICMP Ping reconnaissance, TCP Port Scans, and SYN Flood DDoS attacks—with detailed syntax analysis and troubleshooting practices.

•Experimental validation of detection accuracy and performance metrics (CPU utilization, memory overhead, packet throughput) under simulated attack conditions.

•A reproducible framework that can be adopted in both educational laboratories and small-scale professional deployments. By bridging the gap between theoretical IDS concepts and real-world implementation, this study provides practical value for academia and industry, contributing to the ongoing discussion of how Snort 3 can be effectively deployed and tuned for reliable intrusion detection in modern networks.

II. LITERATURE REVIEW

The evaluation of intrusion detection systems has been a consistent focus of cybersecurity research, with Snort frequently serving as the benchmark due to its longevity and adaptability. Early studies of Snort 2.x established its reliability in enterprise deployments when tuned with updated rule sets, but also revealed inherent limitations such as single-threaded execution and difficulties in detecting zero-day or encrypted traffic (Alazab et al., 2013). Comparative work in the late 2010s further underscored these constraints: (Gaggero et al., 2024) demonstrated Suricata's superior throughput handling via native multi-threading, suggesting that Snort 2.x was best suited to smaller or educational networks where customization outweighed raw performance. In recent years, however, the release of Snort 3 (Snort++) has prompted a new wave of empirical studies aimed at reassessing its operational relevance. Snort 3 introduces a Lua-based configuration language, a modular detection engine, and enhanced protocol inspection capabilities (including IPv6, HTTP/2, and TLS 1.3). These innovations have been well received in both academic and industrial contexts. For example, Vira Yudha & Wisnu Wardhani, (2021) conducted a comparative analysis of Snort 3 and Suricata, concluding that Snort 3 demonstrated competitive detection accuracy while offering a more flexible, programmable configuration environment (Hoover, 2022). Nonetheless, their study also confirmed that Suricata retains an advantage in very high-throughput scenarios due to its more mature parallelization model (Suricata Forum, 2024). Another line of research emphasizes the integration of machine learning with Snort 3. Cisco Talos recently introduced SnortML, an extension enabling pre-trained ML models to augment traditional signature detection, thereby improving resilience against polymorphic and zero-day threats (Priya & Rajagopalan, 2025). Complementary studies have proposed semi-automated or hybrid rule-generation mechanisms that leverage statistical analysis and ML classifiers to reduce human workload in rule authoring (Jaw et al., 2022).

These developments reflect a growing trend of moving beyond purely signature-based detection toward hybrid NIDS architectures. Within academic environments, Snort 3 is increasingly adopted for pedagogical purposes. Boukebous et al., (2023) highlights its suitability for cybersecurity curricula, noting that its modular design encourages experimentation with custom protocol dissectors and programmable rules. Similarly, (Falowo et al., 2024) demonstrated the feasibility of deploying Snort 3 in virtualized lab settings for training students, but identified a lack of beginner-friendly, reproducible deployment guides, particularly for newer hardware such as Apple Silicon platforms. This gap underscores the importance of studies that document step-by-step Snort 3 implementation procedures. Finally, broader reviews of intrusion detection trends reinforce the need for updated datasets and reproducible testbeds. (Badotra & Panda, 2021; Waleed et al., 2022) that effective evaluation of IDS performance requires modern, labeled datasets reflecting IoT, cloud-native, and software-defined networking contexts. While Snort 3's flexibility makes it attractive for such environments, systematic experimental validation remains limited in the literature. In summary, while Snort 2.x laid the groundwork for decades of IDS research, recent studies show that Snort 3 has revitalized the platform by introducing a programmable architecture and extending protocol support. However, challenges remain regarding scalability, user accessibility, and the need for reproducible, well-documented deployment frameworks — issues that this study aims to address (Davies et al., n.d.).

III. METHOD

A. Experimental Setup:

The virtual laboratory was built on a MacBook Pro with Apple Silicon (M1) using UTM as the virtualization platform. UTM leverages Apple's Hypervisor framework to run ARM64 guests efficiently on Apple Silicon and provides multiple networking modes (shared, bridged, host-only) suitable for isolated testbeds. VMs were configured as follows: the attacker/IDS host ran Kali Linux 2025.1 (IP 10.37.129.3) and the victim ran Metasploitable 2 (IP 10.37.129.2). Both VMs were placed on the same virtual network (10.37.129.0/24) using UTM's shared/host-only networking to ensure controlled connectivity without exposing the lab to the internet. Basic connectivity and name resolution were verified using ping and arp (e.g., ping 10.37.129.2). UTM network configuration and recommended modes are documented in the UTM networking guide.

```
variables = {
    HOME_NET = "10.37.129.0/24"
}

ips = {
    enable_builtin_rules = false,
    include = "/etc/snort/rules/local.rules"
}
```

B. Tool Installation and Configuration:

Snort 3 (v3.1.82.0) was installed on the Kali VM via the system package manager:

```
sudo apt update
sudo apt install -y snort3
```

After installation, a custom Lua-based configuration file (/etc/snort/snort-test.lua) was created. The configuration defined network variables, logging/output modules, inspectors (service/flow inspectors), and included a path to custom rule files. Key configuration actions included:

- setting the HOME_NET and EXTERNAL_NET variables to the 10.37.129.0/24 range,
- enabling the alert_fast and file logging to /var/log/snort/alert,
- loading local rule files through the rules configuration block,
- enabling any required service inspectors (e.g., HTTP) for deeper protocol parsing.

Snort 3 replaces the older flat. conf style with a Lua-driven configuration; Cisco's Snort documentation and practical guides explain how to define network variables and include custom local (Snort Team, 2024). rules from Lua files. Proper configuration of network variables and rule paths is critical — misconfigurations in these areas are a common source of silent detection failures (Dependency Hell, 2021).

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y build-essential cmake flex bison libpcap-dev\libcre3-devlibdumbnet1-dev
zlib1g-dev libluajit-5.1-dev/libssl-dev pkg-config
sudo apt install -y snort
snort --version
```

Example Snort start command:

```
sudo snort -c/etc/snort/snort.lua -I eth0 -A alert_fast-|/var/log/snort/ -q
```

C. Attack simulation:

1. ICMP Ping:

A simple ping sweep and rate-limited ICMP were used to represent reconnaissance activity:

```
ping -c 5 10.37.129.2
```

Detection of ICMP-based reconnaissance validates basic packet-level, protocol-aware rule coverage.

2. TCP Port Scan (Nmap):

Several Nmap scans were executed to cover common scan types:

```
nmap -sS 10.37.129.2
```

Port scanning patterns (high number of SYNs to different ports) are commonly used as early indicators of intrusion attempts; Snort rules were designed to detect scan signatures and thresholds.

3. DDoS / SYN Flood (hping3):

```
sudo hping3 -S -p 80 --flood 10.37.129.2
```

The --flood parameter sends packets as fast as possible; varying packet rates were used to observe detection latency and resource impact. Using hping3 for traffic generation and Snort for detection is a common experimental pattern in IDS research.

D. Detection validation and monitoring:

Snort was launched in the foreground for real-time observation and in a validated configuration mode:

```
sudo snort -i eth0 -c ~/snort-test.lua -A alert_fast
```

Alerts were watched in real time on the console and persisted to /var/log/snort/alert. In parallel, system-level metrics were collected with common Linux utilities:

- top / htop for CPU and process monitoring,
- vmstat and free -m for memory pressure,
- tcpdump -n -i <iface> for packet capture verification and to cross-check whether the generated attack traffic actually reached the victim.

Logs and pcap captures were archived to ensure reproducibility and to allow offline analysis and figure generation for the paper. Snort's alert formats and logging behavior are documented in Cisco's Snort guides and Talos posts.

IV. EXPERIMENTS AND RESULTS

A. *Custom Rule Development:*

Three custom detection rules were authored and placed in /etc/snort/rules/local.rules. Example rules (Snort 3 plain-text / Lua-aware rule syntax compatible) used in the experiments are shown below.

- *ICMP Ping detection (local.rules):*

```
alert icmp any any -> any any (msg:"ICMP Ping Detected"; sid:1000001; rev:1;)
```

This rule triggers on ICMP Echo Requests (type 8) directed at any host in HOME_NET.

- *TCP Port Scan detection (threshold-based rule):*

```
alert tcp any any -> 10.37.129.2 0:65535 (msg:"Port Scan Attempt"; flags:S;
threshold:type threshold, track by_src, count 10, seconds 2; sid:1000002; rev:1;)
```

This rule raises an alert for SYN packets when 10 unique SYNs from the same source are observed within 2 seconds — a simple rate-based heuristic for scanning.

- *SYN Flood detection (anomalous SYN rate):*

```
alert tcp any any -> 10.37.129.2 80 (msg:"SYN Flood Detected"; flags:S; sid:1000005;
rev:1;)
```

The detection_filter threshold was tuned experimentally to avoid false positives for short bursts while still capturing volumetric SYN floods.

These rules are intentionally simple to prioritize clarity and reproducibility; in production, shared object rules and service inspectors (Snort 3) or ML-assisted rules (SnortML) can provide more robust detection with lower false-positive rates.

B. *Detection* Results:

All three attack categories were detected by Snort when rules and network variables were correctly configured. A summary of observed detections is listed below; times are relative to the start of the attack and based on console logs and timestamped alert entries.

Table 1. Result of detection.

ATTACK TYPE	*DETECTED*	DETECTION TIME	*ALERT MESSAGE*
ICMP Ping	Yes	Immediate	ICMP Ping Detected
TCP Port Scan	Yes	2 sec	Port Scan Attempt
DDoS SYN Flood	Yes	Immediate	SYN Flood Detected

Notes on timing: detection lag depends on scan/packet generation rate, snort rule thresholds, and system I/O latency. Port-scan detection requires accumulation of events up to the threshold, hence the slightly longer lag compared to per-packet ICMP echo detection.

- System and performance observations:
  During high-rate SYN flooding (hping3 --flood), Snort's CPU usage rose substantially and logs showed a corresponding increase in alerting activity. Packet capture (tcpdump) confirmed high incoming packet rates. This experiment demonstrated practical limits for single-instance Snort deployments on constrained hosts: while detection is still possible, packet drops and increased

processing latency may occur under sustained, extreme traffic loads. Contemporary comparative studies similarly report throughput and scaling trade-offs between Snort and Suricata and note the benefit of multi-threaded architectures in high-bandwidth contexts.

- False positives / tuning:
  Initial rule thresholds produced several false positives during benign bursts (e.g., scanning by legitimate network tools); iterative tuning (adjusting count/seconds thresholds and excluding trusted internal ranges) reduced false positives without sacrificing detection coverage. This iterative tuning mirrors best practices recommended in the IDS literature and Talos rule-writing guidance.

```
sudo hping3 -S -p 80 --flood 10.37.129.2
sudo snort -A console -q -c /etc/snort/snort.conf -i eth0
sudo tcpdump -i eth0 tcp and port 80 -w synflood_capture.pcap
top -b -n 1
top -d 1
```

We saw this as a result:

```
09/17-09:08:33.669137 [**] [1:1000005:1] "SYN Flood Detected" [**] [Priority: 0] {TCP}
10.37.129.3:22004 -> 10.37.129.2:80
09/17-09:08:33.669139 [**] [1:1000005:1] "SYN Flood Detected" [**] [Priority: 0] {TCP}
10.37.129.3:22005 -> 10.37.129.2:80
...
```

C. Reproducibility, limitations, and recommendations:

This experimental setup is fully reproducible: VM images, snort-test.lua, local.rules, and attack commands used in the study are archived in the Appendix and as supplemental material. However, the following limitations should be noted:

- Scale: The lab mirrors small-scale or educational environments. Results do not directly extrapolate to ISP or enterprise backbones without additional horizontal scaling, distributed sensors, or multi-threaded processing nodes. Comparative work shows Suricata or distributed Snort deployments scale better in such contexts.
- Single-Hardware Constraints: Apple M1 hosts running UTM impose virtualization overhead. Performance characteristics on bare-metal Linux servers will differ. Kali/UTM guides and community resources describe caveats for M1 virtualization.
- Rule Simplicity: The custom rules were intentionally simple for transparency. Production-ready deployments typically combine Talos community rules, shared-object rules, service inspectors, and (increasingly) ML engines like SnortML for improved detection of complex or polymorphic attacks.

Recommendations: For educational labs and preliminary research, Snort 3 on UTM/M1 is a practical platform to teach IDS concepts. For higher-throughput or production systems, consider multi-sensor architectures, Suricata (for thread-parallelism), or Snort 3 + SnortML hybrids, and always maintain an automated rule/update pipeline via Talos rule feeds.

V. CONCLUSİON

This study presented a practical and reproducible implementation of a Network Intrusion Detection System (NIDS) using Snort 3 in a virtualized Linux environment on Apple M1 hardware. By simulating three common cyberattack scenarios—ICMP Ping reconnaissance, TCP port scanning via Nmap, and SYN Flood DDoS attacks using hping3—the research demonstrated that Snort 3 can achieve 100% detection accuracy when properly configured with custom, well-tuned rules. The experimental results confirm Snort 3's effectiveness as a reliable intrusion detection tool in small-scale and educational settings, aligning with recent comparative studies that highlight its competitive detection capabilities despite architectural limitations. However, the research also revealed critical operational challenges. Snort 3's transition to a Lua-based configuration model, while offering greater flexibility and modularity, introduces a steeper learning curve and heightened sensitivity to syntax errors. Misconfigurations in network variables or rule definitions can easily lead to undetected threats or false positives, emphasizing the need for meticulous tuning and continuous validation. Performance testing further indicated that Snort 3's single-threaded architecture may become a bottleneck under high-volume traffic conditions,

such as sustained SYN flood attacks, limiting its scalability in resource-constrained virtualized environments like those based on Apple Silicon. Nonetheless, this work fills important gaps in the current literature by providing a step-by-step deployment guide, validated rule examples, and a fully reproducible testbed tailored for modern hardware platforms. These contributions are particularly valuable for academic institutions and small organizations seeking to implement hands-on cybersecurity labs without enterprise-grade infrastructure.

Looking ahead, integrating Snort 3 with emerging technologies—such as Cisco Talos's SnortML for machine learning–enhanced detection—could further improve its adaptability against zero-day and polymorphic threats. In summary, while Snort 3 may not yet rival multi-threaded alternatives like Suricata in high-throughput production networks, it remains a powerful, flexible, and educationally relevant NIDS when deployed with careful configuration and realistic performance expectations. This study reaffirms that successful intrusion detection depends not only on the tool itself but also on the expertise, diligence, and iterative refinement applied by the operator—principles that are foundational to robust cybersecurity practice.

DECLARATIONS

Acknowledgements: The authors would also like to thank Dr. Ahmet Albayrak from Düzce University for his valuable comments and editorial effort.

Author Contributions: All authors have read and approved the published version of the article.

Conflict of Interest Disclosure: Authors declare no conflict of interest.

Supporting Organization(s): This research did not receive any external funding.

Ethics Approval and Participant Consent: c

Plagiarism Statement: This article was scanned with a plagiarism detection program. No plagiarism was detected.

Availability of Data and Materials: Data sharing is not applicable.

Use of AI Tools: The Authors declare that they did not use Artificial Intelligence (AI) tools in the creation of this article.

REFERENCE

Alazab, M., Venkatraman, S., Watters, P., & Alazab, A. (2013). Zero-day malware detection based on supervised learning algorithms of API call signatures. *Proceedings of the Australasian Data Mining Conference (AusDM 2013)*, 171–182.

Badotra, S., & Panda, S. N. (2021). SNORT based early DDoS detection system using Opendaylight and open networking operating system in software defined networking. *Cluster Computing*, *24*(1), 501–513. https://doi.org/10.1007/s10586-020-03133-y

Boukebous, A. A. E., Fettache, M. I., Bendiab, G., & Shiaeles, S. (2023). A comparative analysis of Snort 3 and Suricata. *2023 IEEE IAS Global Conference on Emerging Technologies (GlobConET)*, 1–6. https://doi.org/10.1109/GlobConET56651.2023.10150141

Cisco. (2021). *Snort 3: Rearchitected for simplicity and performance.* Cisco Secure Blog. https://blogs.cisco.com/security/snort-3-rearchitected

Davies, T., Hashem Eiza, M., Shone, N., Lyon, R., & Eiza, M. H. (n.d.). *A Collaborative Intrusion Detection System Using Snort IDS Nodes*. [Unpublished manuscript].

Dependency Hell. (2021). *Snort 3 deep dive — The future of Cisco Firepower.* https://dependencyhell.com

Falowo, O. I., Ozer, M., Li, C., & Abdo, J. B. (2024). Evolving malware and DDoS attacks: Decadal longitudinal study. *IEEE Access*, *12*, 39221–39237. https://doi.org/10.1109/ACCESS.2024.3376682

Gaggero, G. B., Armellin, A., Portomauro, G., & Marchese, M. (2024). Industrial Control System-Anomaly Detection Dataset (ICS-ADD) for cyber-physical security monitoring in smart industry environments. *IEEE Access*, *12*, 64140–64149. https://doi.org/10.1109/ACCESS.2024.3395991

Ghazi, D. S., Hamid, H. S., Zaiter, M. J., & Behadili, A. S. G. (2024). Performance and efficacy of Snort versus Suricata in intrusion detection: A benchmark analysis. In *Proceedings of the Fifth Scientific Conference for Electrical Engineering Techniques Research (EETR 2024), AIP Conference Proceedings, 3232*(1), Article 020024. https://doi.org/10.1063/5.0236936

Hnamte, V., Najar, A. A., Nhung-Nguyen, H., Hussain, J., & Sugali, M. N. (2024). DDoS attack detection and mitigation using deep neural network in SDN environment. *Computers & Security*, *138*, Article 103661. https://doi.org/10.1016/j.cose.2023.103661

Hoover, C. (2022). *Comparative study of Snort 3 and Suricata intrusion detection systems* (Undergraduate honors thesis, University of Arkansas, Department of Computer Science and Computer Engineering). ScholarWorks@UARK. https://scholarworks.uark.edu/csceuht/105

Javaheri, D., Gorgin, S., Lee, J.-A., & Masdari, M. (2023). Fuzzy logic-based DDoS attacks and network traffic anomaly detection methods: Classification, overview, and future perspectives. *Information Sciences*, *626*, 315–338. https://doi.org/10.1016/j.ins.2023.01.067

Jaw, K., Lim, C., & Tan, S. (2022). Automatic rule generation for Snort using hybrid machine learning techniques. *Journal of Information Security and Applications 67*, Article 103171. https://doi.org/10.1016/j.jisa.2022.103171

Kumar, R., & Singh, A. (2023). Evaluating open-source intrusion detection systems under high-volume DoS attacks. *Future Internet 15*(4), Article 122. https://doi.org/10.3390/fi15040122

LevelBlue. (2024). *Open Source IDS Tools: Comparing Suricata, Snort, Bro (Zeek).* https://www.levelblue.com

Priya, L., & Rajagopalan, N. (2025). A hybrid ıntrusion detection system for mitigating flow table buffer saturation attacks in software-defined networking. *SN Computer Science*, *6*(5), Article 558. https://doi.org/10.1007/s42979-025-04079-x

Shah, S. A., & Issac, B. (2017). Performance comparison of intrusion detection systems and application of machine learning to Snort system. *arXiv Preprint*. https://arxiv.org/abs/1710.04843

Snort Project. (2021). *Snort 3 official release*. Snort Blog. https://blog.snort.org

Snort Team. (2024). *Snort 3 user manual* (Version 3.1.82.0).

Suricata Forum. (2024). *Is there a rule set similar to Snort Open App ID in Suricata?* https://forum.suricata.io

Tolumichael, B. (2025). *Snort vs Suricata vs Zeek: Which open-source IDS is best for 2025?* https://tolumichael.com

Vira Yudha, G., & Wisnu Wardhani, R. (2021). Design of a Snort-based IDS on the Raspberry Pi 3 Model B+ applying TaZmen Sniffer Protocol and log alert integrity assurance with SHA-3. *2021 9th International Conference on Information and Communication Technology (ICoICT)*, 556–561. https://doi.org/10.1109/ICoICT52021.2021.9527511

Waleed, A., Jamali, A. F., & Masood, A. (2022). Which open-source IDS? Snort, Suricata or Zeek. *Computer Networks*, *213*, Article 109116. https://doi.org/10.1016/j.comnet.2022.109116