Journal of Internet Services and Applications, 2025, 16:1, doi: 10.5753/jisa.2025.6112
This work is licensed under a Creative Commons Attribution 4.0 International License.

Analysis of Computational Resource Consumption of an Intrusion Detection System Based on Containerized Network Functions Virtualization

Lucas Teles de Oliveira [**Universidade Tecnológica Federal do Paraná** | ***vda.lucas@gmail.com***]
Ana Cristina Barreiras Kochem Vendramin [**Universidade Tecnológica Federal do Paraná** | ***criskochem@utfpr.edu.br***]
Juliana de Santi [**Universidade Tecnológica Federal do Paraná** | ***jsanti@utfpr.edu.br***]
Daniel Fernando Pigatto [**Universidade Tecnológica Federal do Paraná** | ***pigatto@utfpr.edu.br***]

Programa de Pós-Graduação em Computação Aplicada (PPGCA), Universidade Tecnológica Federal do Paraná, Av. Sete de Setembro, 3165, Rebouças, Curitiba, PR, 80230-901, Brazil.

Received: 09 June 2025 • **Accepted:** 28 October 2025 • **Published:** 12 December 2025

Abstract The rapid expansion of global telecommunications networks has driven a continuous increase in Internet adoption, requiring telecom companies to deploy scalable services efficiently to accommodate new users. At the same time, the constant pursuit of cost reduction and improved service delivery has highlighted the need to enhance network function performance. Network Function Virtualization (NFV) addresses these demands by replacing costly, dedicated hardware with virtualized network functions running on virtual machines or containers. This approach enables better resource allocation, scalability, and cost reduction. While traditional virtualization methods can be slow and resource-intensive, container-based solutions, such as those offered by Docker, provide a more lightweight and efficient alternative. By reducing virtualization overhead through kernel sharing, containers significantly streamline the deployment and scalability of NFV-based services. Alongside this evolution, the expansion of online services has brought a surge in cybersecurity threats, highlighting the urgent need for Intrusion Detection Systems (IDS) capable of monitoring traffic patterns and detecting malicious activity in real time. This paper presents a modular testbed framework for NFV-based IDS evaluation, deploying Snort in Docker containers and comparing computational resource consumption against a traditional virtual machine (VM) implementation. The framework enables dynamic instantiation, scalability, and efficient orchestration of IDS components, providing a practical environment to study how different virtualization strategies impact system performance. Specifically, our study i) evaluates the performance of the NFV-IDS running on both a VM and a Docker container, and ii) tests NFV-IDS alongside an Nginx web server under cyberattack. The results provide insights into the viability of containerized NFV for IDS deployment, particularly in environments that demand lightweight, dynamic, and resource-efficient security infrastructures. Furthermore, the framework provides a foundation for future experiments incorporating alternative detection engines, traffic profiles, or virtualization strategies.

Keywords: Intrusion detection systems, network functions virtualization, resource consumption.

1 Introduction

Driven by the continuous expansion of global telecommunications networks, the number of Internet users worldwide reached 5.64 billion by April 2025, representing 68.7% of the global population, according to [DataReportal *et al.*, 2025]. In response to this growing demand, the European Telecommunications Standards Institute (ETSI) [ETSI, 2019] indicates that telecom companies must be able to launch scalable new services and quickly accommodate new users. The ability to automatically scale network functions is one of the primary benefits of Network Function Virtualization (NFV) [Adamuz-Hinojosa *et al.*, 2018]. According to Han *et al.* [2015], NFV is a network technology that enables the replacement of costly, dedicated hardware devices, such as routers, firewalls, load balancers, and intrusion detection systems, with virtualized network functions running on virtual machines or containers. This capability enables more efficient resource allocation, reducing unnecessary computational costs and minimizing the need for additional hardware investments.

While NFV introduces a new paradigm [ETSI, 2019], traditional virtualization processes can be time-consuming, which, in some cases, makes the virtualization of network functions slow and costly. In contrast, Docker [Inc., 2013], a widely used containerization tool today [Susnjara and Smalley, 2024], simplifies this process by allowing the creation, testing, and deployment of applications in isolated environments called containers, thus making the virtualization process easier. As there is no need to create and simulate hardware, the container shares the host system's kernel. An NFV transformed into a Docker image can be instantiated as a container in any desired environment, allowing the application to be used on the developer's computer in the same way that it would be executed on the production server.

Although these technologies significantly improve deployment efficiency and scalability, the increasing reliance on digital services also broadens the attack surface of modern networks. As a result, this growing demand for new online services has created an increasingly favorable environment for cybersecurity threats [Forum, 2025]. To address these challenges, an Intrusion Detection System (IDS) has become an essential component of modern security infrastructures. An IDS is designed to identify and respond to various types of intrusions by continuously monitoring network traffic and analyzing patterns to detect suspicious activities, malicious behaviors, or violations of predefined security policies [Abdulganiyu *et al.*, 2024].

Despite the adoption of NFV by several organizations, the initial vision of offloading middlebox functionality to external, virtualized infrastructures [Sherry *et al.*, 2012] remains only partially realized. Ongoing challenges related to the stability, performance, and scalability of NFV platforms continue to hinder its widespread deployment [Wang *et al.*, 2022].

This paper evaluates a Docker-based NFV implementation of the Snort IDS and compares its resource consumption with that of a traditional VM deployment. The analysis focuses on CPU, memory, and bandwidth utilization under different DDoS attack scenarios. Unlike previous studies, this work contributes a comparative perspective on virtualization models, providing insights into the scalability and resource efficiency of IDS deployments in lightweight environments. Additionally, this study aims to: (i) compare traditional Virtual Machines (VMs) with Docker containers; (ii) evaluate the feasibility of using NFV-IDS in terms of scalability and mobility; and (iii) test NFV-IDS in conjunction with an Nginx web server under attack. The choice of Nginx is motivated by its lower memory consumption compared to Apache and its ability to handle web requests using an event-driven model, whereas Apache relies on a process-driven model.

The main contributions of this work are as follows:

- The design and implementation of a modular testbed framework that enables the deployment and evaluation of IDS components as containerized VNFs within an NFV environment;
- A systematic evaluation of an NFV-based IDS deployed in both containers and virtual machines, highlighting differences in CPU, memory, and bandwidth consumption;
- A detailed analysis of the resource behavior of containerized IDS under different types of DDoS attacks (TCP, UDP, and HTTP);
- A comparison of container and VM environments, showing the advantages of containers in terms of scalability, storage efficiency, and deployment simplicity;
- Insights into how resource consumption patterns can support future IDS research, especially in lightweight and resource-constrained environments.

This paper is organized as follows: Section 2 presents the background on virtualization, NFV, and IDS; discusses several types of denial-of-service attacks; and reviews related work. Section 3 describes the employed NFV-IDS model. Section 4 evaluates the resource consumption of the NFV-IDS during attacks. Finally, Section 5 provides the concluding remarks of the paper.

2 Background

This section presents the background on Virtualization, Network Function Virtualization (NFV), Intrusion Detection Systems (IDS), and some types of service attacks that can target modern networks. NFV has revolutionized the way network services are deployed and managed, enabling greater flexibility, scalability, and cost efficiency. Meanwhile, IDS plays a crucial role in detecting and mitigating malicious activities within virtualized networks. Understanding the different types of attacks is essential for securing these systems. Finally, the section reviews some related works, highlighting previous research on the integration of IDS within NFV environments and the challenges associated with securing virtualized network functions against evolving threats.

2.1 Virtualization

Virtualization is a software layer that creates a virtualized architecture, which can manage computing resources such as hard disks, network interfaces, memory, and processors [van Cleeff *et al.*, 2009]. It provides a consistent interface, decoupling software from the hardware layer and enhancing mobility across hosts with varying computational resources.

Figure 1 illustrates two forms of virtualization: virtual machines and containers. Virtualization through virtual machines is typically associated with the concept of multiple guest operating systems (OS) running on the same physical computer (host) [VMware, 2005]. In contrast, container virtualization does not require recreating the entire system in a virtualized environment [Merkel, 2014]. For example, Linux Containers (LXC) allow the creation of multiple isolated instances of an operating system on a single host, with all instances sharing the same operating system core (kernel). Docker, a widely adopted platform for container-based virtualization, encapsulates applications along with their dependencies into lightweight and isolated execution environments, ensuring consistent behavior across different computing infrastructures while minimizing resource consumption. By eliminating one layer of the virtualization stack, container virtualization simplifies the architecture.

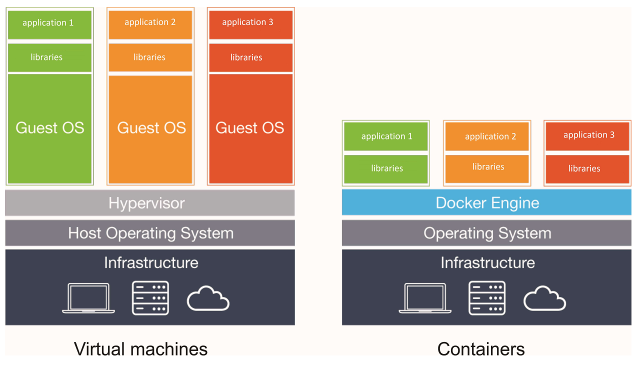

Figure 1. Virtualization on Linux (adapted from [Inc., 2013]).

2.2 Network Functions Virtualization

NFV introduces a new paradigm for the design and operation of telecommunication networks [ETSI, 2016]. It enhances flexibility, shortens development cycles, and facilitates the implementation of new services. By decoupling software from hardware, NFV enables efficient provisioning and resource allocation, particularly for network functions formerly tied to middleboxes [Sherry *et al.*, 2012]. Moreover, it promotes interoperability and open-source adoption, allowing functions to run on diverse hardware platforms. This decoupling reduces dependencies among computing resources and supports the separation of applications from their underlying operating systems.

As an emerging technology, NFV poses several challenges for network operators, including the need to optimize performance, streamline the deployment of network functions, and ensure scalability [Wang *et al.*, 2022; Upadhyay *et al.*, 2024; Çetin *et al.*, 2025]. Despite these challenges, NFV remains a compelling solution due to its ability to support scalable service delivery while reducing computational resource demands. Its practical applications are extensive [ETSI, 2019; BOJOVIC, 2024], encompassing various network components such as switching equipment, mobile network devices, and security devices, including firewalls and IDS systems.

Figure 2 illustrates the layered structure of the NFV framework, highlighting the ability of multiple network functions to share common computational resources. According to the ETSI architecture [ETSI, 2014], the NFV framework is organized into three main domains: i) the NFV Infrastructure (NFVI), which includes the hardware layer, comprising physical resources such as servers, storage systems, and network devices, and the virtualization layer, responsible for abstracting these resources into virtual machines or containers ii) the Virtualized Network Functions (VNF) domain, where traditional network services (e.g., firewalls, load balancers, and IDSs) are implemented as software-based functions; iii) and the management and orchestration (MANO) domain, which oversees the deployment, scaling, and monitoring of VNFs.

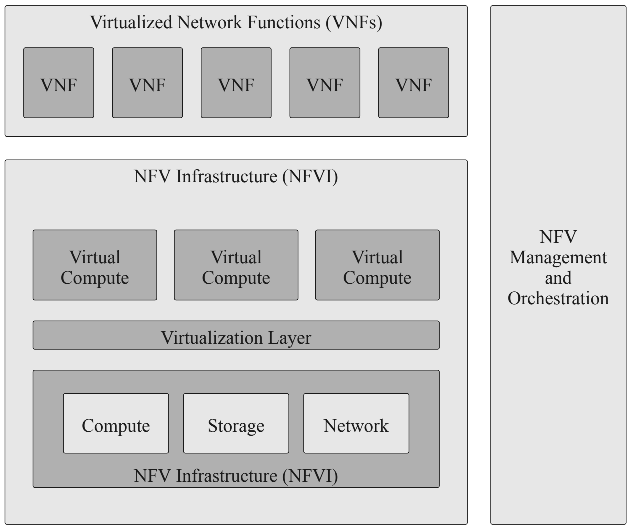
Virtualized Network Functions (VNFs)
VNF
VNF
VNF
VNF
VNF
NFV Infrastructure (NFVI)
Virtual Compute
Virtual Compute
Virtual Compute
Virtualization Layer
Compute
Storage
Network
NFV Infrastructure (NFVI)
NFV Management and Orchestration

Figure 2. ETSI NFV framework [ETSI, 2014].

2.3 Intrusion Detection Systems

An Intrusion Detection System (IDS) silently monitors both the input and output of data, with its primary function being the real-time detection of unauthorized activities within a network or on the system itself [Tiwari *et al.*, 2017]. The concept of IDSs emerged in the mid-1980s, driven by the increasing number of intrusions into systems connected to Advanced Research Projects Agency Network (ARPANET). As cyber-attack techniques evolve at the same pace as the development of new defense technologies, it is essential for an organization's security policies to remain continuously updated [**?**].

The defense systems can be categorized into two models: i) IDS, which are designed to monitor and detect unauthorized access; and ii) Intrusion Prevention Systems (IPS), which not only monitor but also take proactive measures to block unauthorized access [Ashoor and Gore, 2012].

An IDS can be host-based (HIDS), typically installed on a specific computer or server to monitor its activity, or network-based (NIDS), which monitors all incoming and outgoing data traffic across a network or connected devices [Tiwari *et al.*, 2017]. When unauthorized access is detected, the IDS alerts network administrators by sending notifications, signaling potential security breaches within the network.

Beyond their deployment models, IDSs can also be categorized by the techniques they employ to detect malicious activity [Ashoor and Gore, 2012]: signature-based, anomaly-based, and specification-based. A signature-based method detects intrusions by comparing network traffic or system activities to predefined patterns or "signatures" of known attacks. It is highly accurate for detecting known threats but may fail to identify new or unknown attacks. An anomaly-based method establishes a baseline of normal behavior for network traffic or system activities and detects intrusions by identifying deviations from this baseline. While effective at spotting unknown attacks, it may generate a higher number of false positives. A specification-based approach uses predefined rules or specifications that define valid behavior for a system or network. The signature-based approach is particularly relevant to our research, as it tends to generate fewer false positives, which is crucial when analyzing computational resource consumption.

There are several intrusion detection and prevention systems. Snort [Roesch, 1998] is a system that works by monitoring (sniffing) packets based on libpcap libraries, which results in a very lightweight IDS. As open-source software, Snort has gained worldwide recognition as one of the leading intrusion detection and prevention systems for computer networks [Albin and Rowe, 2012]. Snort can be configured to operate in three modes [Snort Project, 2020]:

1. Sniffer mode, which reads network packets and displays them to the user in a continuous stream on the console;
2. Packet Logger mode logs packets to disk;
3. NIDS Mode, which performs network traffic detection and analysis. This is the most complex and configurable mode.

Other examples of IDS include [AT&T, 2019]: i) Suricata, developed by the Open Information Security Foundation (OISF), provides robust traffic monitoring and anomaly detection capabilities, making it well-suited for deployment in high-performance network environments; ii) Zeek (formerly known as Bro-IDS), which differs slightly from Snort and Suricata in its approach; iii) Open Source Security Event Correlator (OSSEC), a full-featured open-source HIDS with client/server-based management; and iv) Samhain, which operates similarly to OSSEC with a client/server architecture, but stands out by offering easy deployment and data export through its web console.

In this work, we consider IDS-Snort in sniffer mode, analyzing all packets passing through the network.

2.4 Denial of Service Attacks

Denial of Service (DoS) attacks aim to disrupt the availability of a specific service. Over time, the Distributed DoS (DDoS) attack emerged, leveraging multiple distributed sources to amplify the attack. DoS attacks utilize a wide variety of methods to make the targeted service unavailable [Sieklik *et al.*, 2016]. These attacks can take several forms, including:

- TCP (Transmission Control Protocol) SYN Flood;
- UDP (User Datagram Protocol) Flood;
- ICMP (Internet Control Message Protocol) Flood;
- Smurf Attack;
- HTTP (Hypertext Transfer Protocol) Flood.

The targets of these attacks are, in most cases, devices that use network protocols such as HTTP, TCP, UDP, and ICMP. Many of these attacks exploit vulnerabilities and flaws in telecommunications networks and web services. The impact of an attack, whether a successful Smurf attack or an HTTP Flood, can last for hours or even days, resulting in lost revenue, frustrated targets, and potential information theft [Yusof *et al.*, 2017].

TCP SYN and UDP Floods are Layer 4 attacks that exhaust server resources [Yusof *et al.*, 2017]. SYN Floods send spoofed requests without completing handshakes, creating half-open connections. UDP Floods target random ports, triggering ICMP replies that consume processing power and bandwidth, potentially degrading performance or causing outages.

At the network layer, the ICMP Flooding involves sending an excessive number of Echo Request messages to a target, overwhelming it with replies and disrupting legitimate traffic [Yusof *et al.*, 2017]. The Smurf attack similarly exploits ICMP by using broadcast replies to flood the victim, causing delays, failures, or crashes [Yusof *et al.*, 2017].

At the application layer, the HTTP Flood attack exploits HTTP requests, primarily GET and POST methods, to target a web server [Tripathi *et al.*, 2016]. GET requests typically retrieve static resources such as text and images, while POST requests are used to send data to the server, often interacting with dynamically generated content. In an HTTP GET attack, multiple computers coordinate to send a large number of requests to the target server, aiming to overwhelm it and cause a denial of service for legitimate traffic. This process is illustrated in Figure 3, which compares the execution flow of a TCP Flood (left side) with that of an HTTP Flood (right side). While HTTP Flood attacks are generally easier to deploy and scale, they require more server-side processing and can be more challenging to mitigate compared to lower-layer attacks.

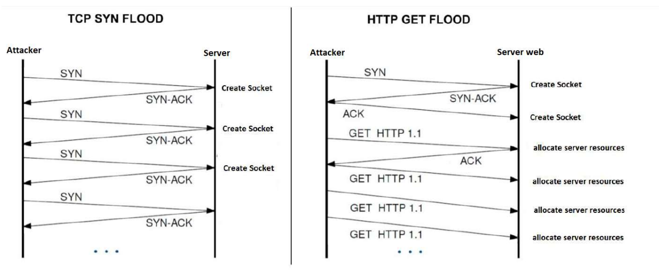

Figure 3. Execution flow of TCP Flood and HTTP Flood attacks (adapted from [Julienne, 2016]).

2.5 Related Work

This section discusses various studies related to Intrusion Detection Systems (IDS) and virtualized technologies, focusing on their applications in improving network security and the scalability and performance of network infrastructures.

Abdulganiyu *et al.* [2024] explore the increasing importance of IDSs in securing sensitive online data in the face of rising cyber threats. Unlike many previous reviews that primarily focused on anomaly-based IDS, the research provides a systematic analysis of signature-based and hybrid IDS approaches as well. By following the Preferred Reporting Items for Systematic Reviews and Meta-Analyses guidelines, the study offers a comprehensive evaluation of IDS methodologies, identifying research gaps and unresolved challenges. The findings highlight future research directions with high potential impact, aiming to enhance IDS models for improved cybersecurity.

Rangisetti [2024a] highlights the virtualization of 5G core network components, focusing on the separation of control and user plane functions to address the specific demands of 5G applications. Additionally, the study provides a step-by-step guide on deploying VNFs using the Docker platform, demonstrating how virtualized networking services can be effectively implemented in practice.

Rangisetti [2024b] explores the significance of NFV and its impact on the flexible management of complex infrastructures, such as cloud environments and telecom core networks. The study highlights that virtualization technologies enable operators to automate various tasks, including the deployment of software, network functions, and provisioning of environments to support multiple customer applications on shared infrastructure. By virtualizing key network functions such as IP addressing, naming services, proxy servers, and load balancing, the author claims that NFV facilitates easier management and control of essential network operations. The author also emphasizes the role of NFV architecture in transforming traditional network functions into virtualized services, thereby enhancing efficiency and scalability.

Kurek *et al.* [2024] explore the transition of NFV technology from the experimental phase to becoming a standard for modern telecommunications networks. The authors predict that, in the near future, all network services will be implemented in software based on cloud-native architectures. The study specifically evaluates the performance of a firewall service based on IncludeOS unikernels, demonstrating that IncludeOS unikernels achieve promising results compared to Ubuntu-based virtual machines and containers. The evaluation is based on a series of experiments and benchmarks designed to assess how the performance of the firewall service is influenced by the number of firewall rules.

Shayegan and Damghanian [2024] address the prevention of Distributed Denial-of-Service (DDoS) attacks in enterprise networks using NFV and Software-Defined Networking (SDN). DDoS attacks aim to disrupt network services by flooding the system with fake requests. The integration of NFV introduces a flexible and dynamic network design that helps mitigate such threats. The study employs the Moving Target Defense (MTD) technique, which dynamically alters routing and the location of detection packets, preventing attackers from targeting the network's real topology. A key innovation of this work is the selection of MTD types based on the processing resources available in overlay networks. The results demonstrate that this approach effectively conserves resources and reduces the time required for network packet verification.

Mauricio and Rubinstein [2023] propose a security architecture based on NFV for automated detection and mitigation of malware in Web applications. The solution includes an NFV Security Controller that orchestrates chains of Virtual Security Functions (VSFs) in response to attack alerts, an IDS-VSF for detecting malicious traffic, a Web Application Firewall (WAF-VSF) that enforces dynamic blocking rules, a vulnerability scanner that proactively defines mitigation actions through periodic scans, and a mechanism for removing obsolete WAF rules to enhance performance. The architecture is implemented on the Open Platform for NFV (OPNFV).

Tikhe and Patheja [2023] present a survey on DDoS mitigation strategies with a focus on NFV. They discuss the severity of DDoS attacks and the limitations of mitigation approaches based on hardware middleboxes and cloud-based protection. The paper highlights how NFV enables flexible, cost-effective, and scalable deployment of virtualized network functions for DDoS defense. It further reviews several NFV-driven strategies to manage attack traffic volume, concluding that NFV provides clear advantages over conventional approaches, although many solutions remain limited in scope or restricted to specific types of DDoS attacks.

Wang [2023] addresses three main NFV deployment challenges: Network Function (NF) execution, interconnection, and task scheduling. To optimize NF execution, the author proposes Lemur, a cross-platform NFV framework that efficiently places and executes NF chains across heterogeneous hardware while meeting Service Level Objectives (SLOs). For NF interconnection, the study introduces Quadrant, a cloud-deployable NFV platform that enhances packet isolation and auto-scaling, improving per-core throughput by up to 2.31× compared to existing NFV systems. Lastly, for task scheduling, the Ironside system is developed to mitigate latency spikes caused by bursty traffic, achieving sub-millisecond p99 latency SLOs with a comparable number of cores. The proposed solutions demonstrate significant advancements in NFV scalability and efficiency, making NFV more viable for large-scale, high-performance cloud environments.

Zahoor *et al*. [2023] examine the role of virtualization technologies in the deployment of Next-Generation Wireless Networks (NGWN), particularly in the context of 5G networks. The study highlights the significance of NFV in addressing key challenges such as scalability, adaptability, and reliability. It compares hypervisors, cloud platforms, and container-based virtualization, focusing on their efficiency in handling network operations at the 5G edge, where low latency and resource optimization are critical. Experimental results demonstrate that containers outperform hypervisor-based infrastructures and cloud platforms in terms of latency and network throughput, though they require higher virtualized processor usage.

Fadhlillah *et al*. [2021] conduct attack tests using the Low Orbit Ion Cannon (LOIC), Tor's Hammer (Torshammer), and XerXeS tools, both with and without an IDS in the test scenario. Additionally, a Web and a File Transfer Protocol (FTP) server were set up to carry out the experiments. The IDS successfully detected the incoming attacks. It is important to note that the primary objective was to assess the efficiency of the IDS tool and analyze the consumption of available resources, such as processing power, memory, and the number of packets detected by the IDS.

Gebert *et al*. [2017] investigate scenarios where NFV could be practically applied in real-world settings by comparing packet processing times as the load increases (measured by TCP connections) between a commercial hardware-based firewall and its virtualized counterpart on VMware.

Brumen and Legvart [2016] compare the Snort and Suricata IDSs in terms of their performance. Experiments were conducted to evaluate the computational resource consumption and detection capabilities of each IDS, with tests performed on both Linux and Windows operating systems. The results indicated that Suricata consumed more CPU and RAM than Snort, regardless of the operating system. However, when comparing Linux and Windows, the number of dropped packets was lower on Linux for five of the six simulated attacks. The authors concluded that the Windows operating system, when tested with large amounts of data, exhibited a higher rate of packet loss. Another use case involves a framework designed to provide converged voice and data over a 4G (Long-Term Evolution - LTE) network, known as the Evolved Packet Core (EPC). In this case, the NFV-based implementation of EPC becomes particularly interesting for both academia and industry. By leveraging NFV, EPC has the potential to address scalability issues quickly.

NFV-based IDS can suffer from performance degradation caused by virtualization overhead, and several techniques have been explored to address this issue [Fei *et al*., 2020]. Hardware-assisted approaches leverage GPU (Graphics Processing Unit) acceleration to speed up packet inspection, while data-plane technologies such as DPDK (Data Plane Development Kit), XDP (eXpress Data Path), or SR-IOV (Single Root I/O Virtualization) reduce kernel involvement and latency. System-level optimizations, including huge pages, NUMA (Non-Uniform Memory Access) awareness, and CPU pinning, further enhance the performance of a virtual network function. Architectural solutions based on microservices have also been proposed to improve scalability and resource utilization in IDS deployments. These approaches are complementary to our container-based solutions, which focus on lightweight orchestration and deployment flexibility, and future research could combine such techniques to further enhance IDS performance.

Table 1 presents an overview of the previously discussed studies, highlighting how our proposed approach compares to and differentiates itself from current solutions in the literature.

In summary, while prior studies have explored NFV for firewalls, malware detection, or general IDS benchmarking, they often lacked a detailed evaluation of container-based IDS performance under DDoS scenarios. Our work complements these studies by focusing specifically on computational resource consumption (CPU, memory, and bandwidth) when deploying Snort-based IDS in containerized environments, and by comparing the results with VM-based virtualization. This focus provides practical insights for operators considering container technologies for security deployments.

3 The NFV-IDS approach

This section describes the NFV-based deployment of Snort IDS in Docker containers and virtual machines, which serves as the basis for our comparative evaluation. The main objective is to evaluate the computational overhead introduced by different virtualization environments when running the same IDS function.

Figure 4 illustrates the NFV-IDS architecture. The Docker host server is responsible for storing three containers:

- The first container runs a web server, which provides the target application. The web server was developed using Nginx since it allows the NFV-IDS to perform its function, running on both the VM and the container. It hosts a website that simulates a personal website;
- The second container contains the NFV-IDS, which monitors access to the eth0 network interface using Snort. Snort operates in sniffer mode, analyzing all packets passing through the network, relying exclusively on the official Snort community rules [Snort Project, 2020], with no custom modifications to detection logic or rule sets. This ensures comparability between container-based and VM-based executions. This applies to both its execution in a traditional virtual machine (VM) and in a container;
- A third container runs Portainer, a platform that facilitates the management and orchestration of containers [Cresswell and Lapenna, 2017].

For the VM-based scenario, Snort was deployed in a traditional virtual machine running Ubuntu 18.04.5 LTS. VM resource usage was monitored using Htop.

Traffic from the Internet first passes through a switch, which forwards packets to the web server. A mirror port duplicates this traffic to the IDS container, allowing Snort to analyze packets without interfering with the service. This design isolates the IDS from the application while ensuring visibility of both legitimate and malicious traffic, while also avoiding additional latency for end users.

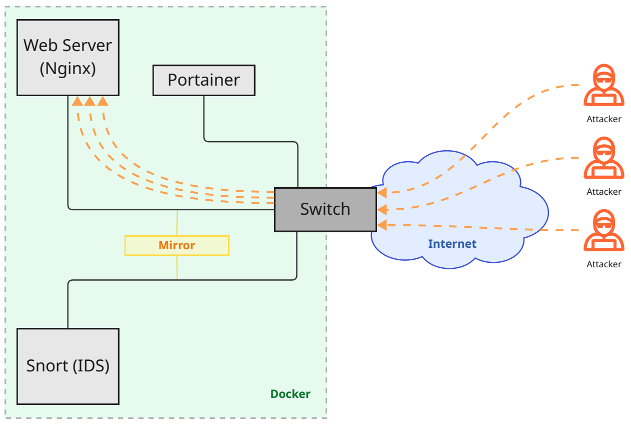

Figure 4. The NFV-IDS Architecture.

The DDoS attacks were generated using Hping3 (TCP and UDP) and LOIC (HTTP) from up to three attacker nodes, with traffic parameters aligned to commonly used settings in related works. This setup is intentionally lightweight, focusing on isolating the impact of the virtualization environment on system performance. By keeping the IDS configuration constant across environments, observed differences in computational resource consumption (CPU, memory, and bandwidth) can be directly attributed to the underlying virtualization technology.

4 Resource consumption analysis

We compare the traditional VM and Docker container as virtualization models for the proposed NFV-IDS (Section 4.1), and also analyze the behavior of the NFV-IDS using the container (Section 4.2).

The analysis focuses specifically on DoS attacks. These attacks were selected because they are among the most resource-intensive and directly impact CPU, memory, and bandwidth consumption, which are the focus of this study.

Table 2 shows the hardware used in the experiments. Each hardware component had 8GB of RAM and a Gigabit interface card.

Each test was repeated 15 times for five minutes per session, and resource monitoring was performed using Portainer for containers and Htop for VMs. The concept of DDoS attack amplification [Oliveira *et al.*, 2020] was also considered. Attack traffic was generated using Hping3 (TCP and UDP) and LOIC (HTTP), with parameters aligned with common settings in related work. Default Hping3 packet sizes were adopted unless otherwise noted, and transmission/reception rates were tuned to reproduce realistic traffic volumes without overwhelming the attacker host. For HTTP tests, the number of threads was varied between 10 and 25, as lower values generated negligible load while higher values caused packet loss at the attacker side. This configuration allowed us to evaluate the system under realistic and controlled attack scenarios.

4.1 Virtualization performance

To evaluate the performance of the NFV-IDS running in both a traditional VM and a Docker container, the following metrics related to computational resource consumption were used:

Table 1. Comparison of Related Works.

Authors	**Focus**	**Technology/Approach**	**Key Contributions/Findings**
Abdulganiyu et al. [2024]	Systematic review of IDS models	Signature-based and hybrid IDSs	Identifies gaps and suggests future directions for effective IDS design
Rangisetti [2024a]	VNF deployment in 5G networks	Docker-based VNF implementation	Demonstrates practical deployment of VNFs and separation of control/user planes
Rangisetti [2024b]	NFV management in telecom/cloud environments	Virtualization of IP services, automation	NFV simplifies provisioning and supports scalability and automation
Kurek et al. [2024]	Performance of firewall services using unikernels	IncludeOS vs. Ubuntu VMs and containers	Unikernels outperform VMs in lightweight, rule-based firewall scenarios
Shayegan and Damghanian [2024]	DDoS prevention in SDN/NFV-based networks	Moving Target Defense (MTD) strategy	Adapts routing based on resource availability; reduces packet verification time
Mauricio and Rubinstein [2023]	Malware detection in Web apps	NFV-based architecture with dynamic VSFs	Orchestrates WAF and IDS dynamically using Open Platform for NFV (OPNFV)
Tikhe and Patheja [2023]	Survey on DDoS mitigation via NFV	Review of techniques	Summarizes NFV-based DDoS countermeasures and efficiencies
Wang [2023]	NFV optimization for cloud scalability	Lemur, Quadrant, Ironside platforms	Improves NF chaining, latency, and interconnect; scalable NFV deployment
Zahoor et al. [2023]	NFV in 5G/NGWN	Hypervisor, cloud, and container comparison	Containers offer lower latency and better throughput, though need more vCPU
Fadhlillah et al. [2021]	IDS effectiveness under DDoS attacks	LOIC, XerXeS tools with IDS	Evaluates IDS under attack scenarios; analyzes resource usage and detection
Gebert et al. [2017]	NFV vs. hardware firewall performance	VMware-based VNF comparison	NFV shows viable performance compared to hardware under increasing load
Brumen and Legvart [2016]	Snort vs. Suricata performance	Performance benchmarking on Linux/Windows	Snort uses less CPU/RAM; fewer dropped packets on Linux
This work	Resource-efficient IDS using NFV	Container-based NFV-IDS using Docker + Snort	Proposes a scalable and lightweight IDS using Docker containers; evaluates CPU, memory, and bandwidth under DDoS attacks

Table 2. Hardware details.

Node	OS version	Processor
Docker Computer	Ubuntu 18.04.5 LTS	i5-8250U
Attacker Computer 1	Windows 10	i5-7500
Attacker Computer 2	Windows 10	i5-7500
Attacker Computer 3	Windows 10	i5-7500

data storage, processing, and memory usage. The goal was to identify the challenges and issues associated with running the NFV-IDS in containers, thereby guiding the subsequent experiments (Section 4.2).

Both a local and a distributed scenario were considered. In both scenarios and virtualization models, the NFV-IDS and the web server shared the same hardware. The three attackers (Figure 4) shared the same hardware for the local scenario and used independent hardware for the distributed scenario. In terms of the connection between the attackers and the server, a point-to-point connection was considered for the local scenario, and a star connection was used for the distributed scenario. A processing core was allocated for both the web server and the NFV-IDS. Moreover, 1024MB of memory was required for each one (web server and NFV-IDS) when using container virtualization, while 2048 MB of memory was necessary for virtualization using a traditional VM. This difference demonstrates how containers can improve resource utilization compared to VMs.

The combinations of $i)$ type of DoS attack (UDP Flood and SYN Flood), $ii)$ the packet size (15 KB and 1500 KB), and $iii)$ the type of environment (local or distributed) were considered in the experiments.

The units reported (KB) refer to the size of the packets after amplification in a controlled DDoS test scenario, not the original network packets. In the experiments, the actual packet sizes were amplified by factors of 100 and 1000, based on the MTU limit, in order to observe measurable effects on computational resource consumption. This amplification is necessary to evaluate resource usage under high-volume traffic conditions in a controlled environment, as discussed in [Colella and Colombini, 2014] and [Kuhrer *et al.*, 2014].

It is important to note that the analysis of memory and processing consumption was conducted independently, as it was possible to monitor consumption in real time, both in Docker containers and in the traditional VM. Portainer was used to monitor the containers, while operating system resource monitors, such as Htop, were used to monitor the VM.

During the configuration and execution of the experiments, as noted by Mijumbi *et al.* [2016], we found that the NFV-IDS container running on Docker offers mobility, making it easy to update and configure even while running. Moreover, as shown in Figure 5, the NFV-IDS with container-based virtualization requires significantly less storage memory compared to virtualization with VMs when creating new images with different resources. This difference arises from the installation and configuration process of the NFV-IDS in a traditional VM, which takes more time due to the need to emulate hardware and the separation between the installation and the operating system that will run the function.

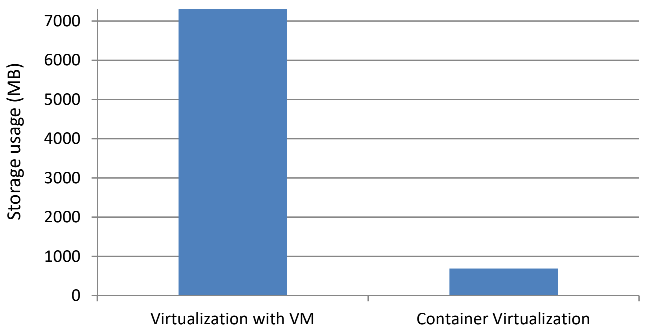

Figure 5. Data storage.

Figure 6 shows the NFV-IDS processing consumption when varying the type of attack (SYN Flood (TCP) and UDP Flood), the environment (local and distributed), and the packet size (1500 KB and 15 KB). For SYN Flood attacks, the processing consumption of NFV-IDS running in a container was greater than that of the Traditional VM in both local and distributed scenarios. The difference reached up to 21.31% (for the local scenario with a packet size of 15 KB). The higher CPU usage observed in containers under TCP SYN Flood attacks is a consequence of their kernel-sharing model. Since containers process packets directly through the host networking stack (a shorter path without the hypervisor or guest kernel) they avoid the virtualization overhead of VMs. This efficiency allows a higher packet rate to reach the IDS, which increases its workload and leads to CPU saturation more quickly.

In contrast, for UDP Flood attacks, the difference in processing consumption between NFV-IDS running on the container and the Traditional VM was lower, as there is no need to wait for a server response.

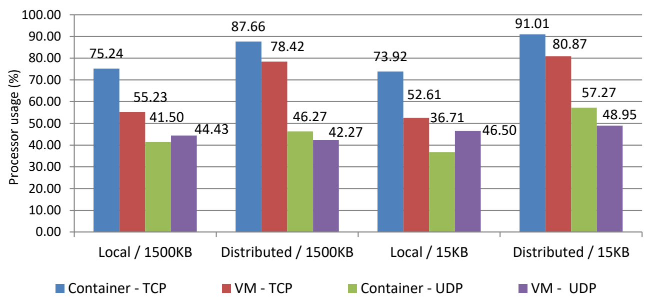

Figure 6. Processing consumption.

Figure 7 presents the memory consumption of NFV-IDS as a function of the type of attack, environment, and packet size. The NFV-IDS using a Docker container exhibited lower memory consumption compared to the traditional VM for both SYN Flood and UDP Flood attacks, regardless of the environment type or packet size. For the SYN Flood attack, the memory consumption of the container did not exceed 600MB, while for the UDP Flood attack, the consumption remained below 300MB. On the other hand, during the SYN Flood attack, the memory consumption of the VM exceeded 1700MB, while during the UDP Flood attack, it exceeded 1400MB.

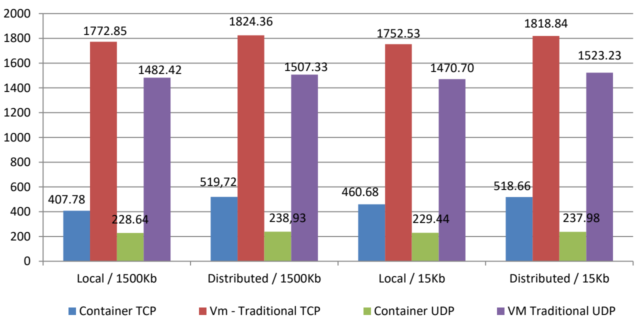

Figure 7. Memory consumption.

Through the analysis of resource consumption for the proposed NFV-IDS implemented in both traditional VMs and containers (Docker), other characteristics of the proposed virtualization model were observed. During the experiments, it was noted that containers (Docker) tend to be faster to deploy, as only the applications need to be started, rather than the entire operating system. Another feature observed with Docker was that roles could be quickly moved from deployment to production using management tools like Portainer.

4.2 Container performance

The results presented in Section 4.1 were instrumental in comparing virtualization technologies, which drove the choice of using containers for the proposed NFV-IDS. Furthermore, given the significant rise in HTTP-based DDoS attacks, illustrated by a 118% year-over-year increase within the Cloudflare network during the first quarter of 2025 [Yoachimik and Pacheco, 2025], we also evaluate the resource consumption of the proposed NFV-IDS deployed in Docker containers under such attack scenarios.

Since the Hping3 tool does not generate HTTP-based attacks, the LOIC tool [Imperva, 2014] is used in this section to generate DDoS attacks. LOIC comes with predefined packet size settings of 1448 bytes, but it allows users to modify the attack speed (faster or slower), the number of threads used, and the type of attack (TCP, UDP, or HTTP).

The experiments considered in this section take into account three factors: i) the type of attack (HTTP, UDP, or TCP); ii) the speed of requests (quantity of requests by unit of time); iii) and the number of threads. During the parameter definition process, it was observed that above 25 threads, the attacker began to lose packets. It was also observed that with fewer than 10 threads, the processing consumption remained at 0%. Therefore, the defined maximum and minimum thresholds for the number of threads are 25 and 10, respectively. As a result, 12 different combinations of experiments were determined (as shown in Figure 8, x-axis). The metrics analyzed were processing, memory, and bandwidth consumption.

Figure 8 shows the average number of requests for each combination per attacker. TCP/Faster/25 attacks generated the highest number of requests, which can be attributed to the attacker's behavior of sending incomplete requests before the connection times out, at which point it sends another request. Similarly, with UDP/Faster/25, the high number of requests is due to the nature of the UDP protocol, which, when sending data packets, does not guarantee that they will arrive correctly.

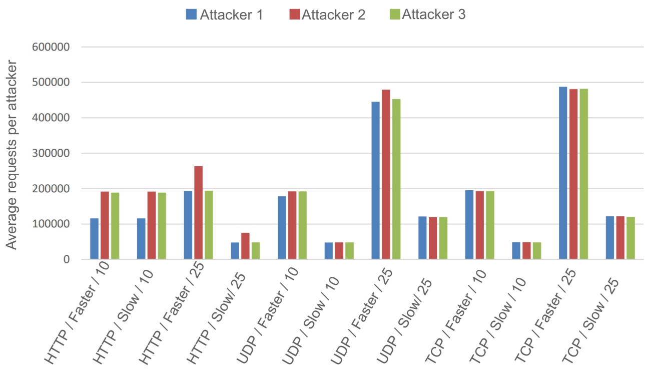

Figure 8. Attack requests.

Table 3 presents the processing (second column) and memory consumption (third column), and their confidence interval variation, for the Nginx web server under HTTP, UDP and TCP attacks. To facilitate comparison between the three types of attacks, processing and memory consumption values are also presented in Figures 9 and 10, respectively.

Regarding processing consumption (Table 3 and Figure 9), during the TCP attack, it remained below 20%, which is considered low but still sufficient to affect the web server. The "HTTP/Faster/25" combination resulted in the highest processing consumption, reaching 100.13%. This high usage is associated with the number of threads and the fact that requests target both the website and its subpages, exploiting the web server's timeout property, which is the maximum duration a request remains in the application's buffer before being served. It is important to note that the reported processing consumption slightly exceeds 100%. This occurs because monitoring tools report usage relative to a single logical CPU. Since the testbed processor has multiple cores with hyperthreading, values above 100% indicate that the IDS and web server together consumed more than the equivalent of one logical core during execution. The UDP combinations ("Faster/10", "Slow/10", and "Slow/25") showed the lowest processing consumption, reaching only 0.02%. Since the UDP attack does not require the web server to respond to requests, CPU usage remains largely unaffected.

Table 3. Resource consumption (Web Server container).

	Processing (%)	Memory(MB)
HTTP		
Faster/10	59.58 (+-0.80)	76.95 (+-0.83)
Slow/10	22.83 (+-0.57)	33.58 (+-0.95)
Faster/25	100.13 (+-0.72)	95.07 (+-0.62)
Slow/25	41.63 (+-0.69)	51.29 (+-0.58)
UDP		
Faster/10	0.02 (+-0.00)	5.11 (+-0.05)
Slow/10	0.02 (+-0.00)	5.07 (+-0.07)
Faster/25	0.03 (+-0.01)	5.23 (+-0.06)
Slow/25	0.02 (+-0.00)	5.03 (+-0.17)
TCP		
Faster/10	13.95 (+-0.36)	9.33 (+-0.04)
Slow/10	1.21 (+-0.09)	5.29 (+-0.05)
Faster/25	19.92 (+-1.30)	9.48 (+-0.05)
Slow/25	11.92 (+-0.32)	5.59 (+-0.06)

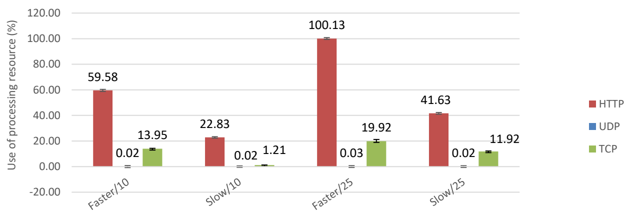

Figure 9. Processing consumption (Web Server container).

Regarding memory consumption (Table 3 and Figure 10), the TCP combinations resulted in memory usage of less than 10 MB, which is under 1% of the total 1024 MB available. Additionally, memory consumption for UDP remained stable, indicating that the web server was not significantly affected. In contrast, the HTTP combinations showed the highest memory consumption compared to the UDP and TCP combinations. The memory usage reached up to 95.07 MB for the "HTTP/Faster/25" combination, which is about 9% of the available 1024 MB.

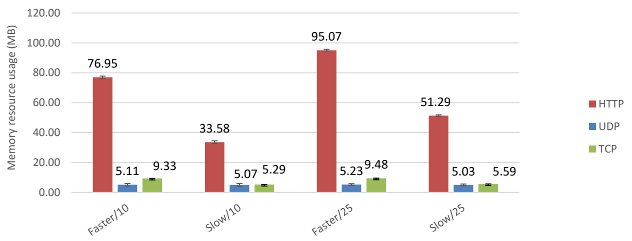

Figure 10. Memory resource consumption (Web Server container).

Figure 11 shows the memory consumption of HTTP combinations over time. In the "HTTP/Faster/25" combination, memory usage increased by approximately 11.2 MB per minute. Consequently, an attack that fragments and spoofs HTTP requests, gradually consuming more memory minute by minute, becomes harder to detect. This can lead to a RAM bottleneck, potentially affecting availability within hours without being noticed.

Figure 12 illustrates the network bandwidth consumption (in Gigabytes, GB), considering the impact of the attack type,

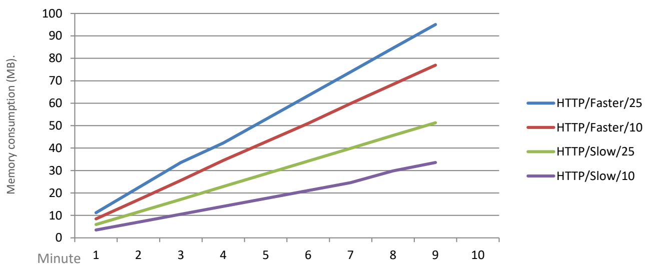

Figure 11. HTTP memory consumption.

number of requests, and available threads. The data were collected from the network interface shared between the NFV-IDS and the web server. The "UDP/Slow/10" combination resulted in the lowest network bandwidth consumption, while HTTP-based attacks caused the highest bandwidth usage, reaching 2.5GB in the "HTTP/Faster/25" combination. It was also observed that three attackers do not generate enough traffic to cause a network bandwidth bottleneck.

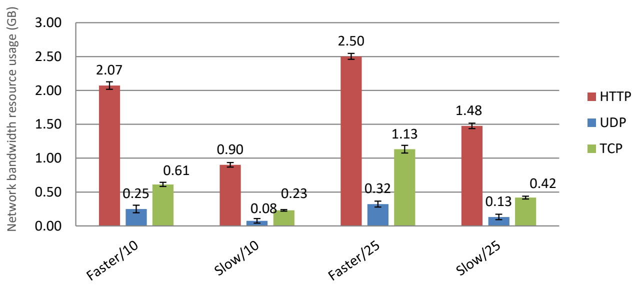

Figure 12. Network bandwidth consumption.

A characteristic observed when using LOIC as the attack tool in the experiments is that network bandwidth consumption remains low due to the small size of the packets. However, the number of requests is high, which aims to stress the processing capacity and RAM consumption.

Table 4. Consumption resources of the NFV-IDS container.

	Processing (%)	Memory(MB)
HTTP		
Faster/10	12.51 (+-0.33)	341.59 (+-0.55)
Slow/10	0.14 (+-0.01)	248.16 (+-2.14)
Faster/25	17.01 (+-0.28)	377.2 (+-3.08)
Slow/25	7.27 (+-0.13)	253.01 (+-3.7)
UDP		
Faster/10	0.67 (+-0.02)	115.75 (+-0.41)
Slow/10	0.63 (+-0.02)	110.27 (+-1.21)
Faster/25	1.57 (+-0.07)	129.07 (+-0.36)
Slow/25	0.77 (+-0.03)	111.51 (+-1.96)
TCP		
Faster/10	12.47 (+-0.25)	118.93 (+-0.86)
Slow/10	10.88 (+-0.46)	120.25 (+-0.67)
Faster/25	36.45 (+-0.45)	247.43 (+-0.20)
Slow/25	5.91 (+-0.10)	125.65 (+-0.38)

Table 4 shows the processing (second column) and memory consumption (third column), and their confidence interval variation, for the NFV-IDS container under HTTP, UDP and TCP attacks. Processing consumption ranged from a minimum of 0.14% CPU for the "HTTP/Slow/10" combination to a maximum of 36.45% for "TCP/Faster/25". This difference reflects the influence of attack type, as the NFV-IDS network function experiences a higher load during TCP attacks due to the nature of the incoming requests.

The amount of UDP and TCP requests resulted in lower memory consumption by the NFV-IDS (Table 4, third column), with the exception of the "TCP/Faster/25" combination, which reached 247.43MB. On the other hand, HTTP requests led to increased memory consumption across all evaluated combinations. As shown in the table, the NFV-IDS container consumed an average of 341.59 MB and 377.2 MB of memory for the "HTTP/Faster/10" and "HTTP/Faster/25" combinations, respectively, representing approximately 34% and 38% of the total available 1024 MB.

4.3 Discussion

During the experiments, the frequency of the attacks did not significantly impact messaging overhead or interfere with the regular data flow. Instead, the main variations in resource consumption were associated with the protocol type and the number of threads rather than with the frequency of attack packets.

Building on this observation, the analysis indicated a correlation between bandwidth consumption and memory usage, particularly under HTTP traffic, where higher request rates resulted in increased memory utilization, as the IDS buffers more packets and maintains more state information. Although a deeper statistical investigation is beyond the scope of this study, this relationship is relevant for understanding patterns of resource saturation.

The experiments further demonstrated that HTTP-based DDoS attacks impose substantially higher resource demands compared to TCP and UDP floods. This behavior aligns with expectations, since HTTP operates at the application layer and requires managing multiple concurrent requests, whereas UDP is connectionless and incurs less processing overhead. Consequently, the type of protocol has a significant impact on IDS performance and resource requirements.

These findings also highlight a practical trade-off between isolation and efficiency: while VMs provide stronger separation from the host OS, containers enable faster deployment and lower memory consumption. However, under TCP flood scenarios, containerized IDS instances may saturate CPU resources more rapidly, underscoring the importance of careful resource tuning and workload-aware deployment strategies.

In summary, the container-based NFV-IDS demonstrated stable performance across TCP, UDP, and HTTP scenarios. CPU and memory consumption were directly influenced by the type of protocol, the request rate, and the number of threads. HTTP traffic emerged as the most resource-intensive, especially in memory usage, confirming its higher complexity compared to TCP and UDP. While three attackers were sufficient to highlight these patterns, we acknowledge that larger-scale scenarios could further stress system resources. These results reinforce the viability of containers as a lightweight and scalable option for deploying IDS in modern infrastructures.

From a security perspective, virtualization approaches also differ. Virtual machines generally offer stronger protection due to their higher degree of isolation, as each VM runs a complete guest operating system on top of the hypervisor.

This separation reduces the risk of host compromise if the IDS is attacked. Containers, in contrast, share the host kernel, which can enlarge the attack surface in the event of kernel vulnerabilities. However, containers provide advantages in terms of rapid updates, easier redeployment, and operational agility, which can enhance resilience in practice. Thus, while VMs provide stronger isolation, containers balance efficiency and responsiveness, making them attractive for IDS scenarios where lightweight deployment and fast recovery are essential.

5 Conclusions

This paper evaluated the deployment of the Snort IDS within an NFV framework, comparing its behavior in containerized and VM-based environments. Rather than introducing a novel IDS, our contribution lies in the comparative analysis of computational resource consumption (CPU, memory, and bandwidth) under different DDoS scenarios. The results highlight the trade-offs between virtualization models and provide insights for selecting lightweight, scalable, and resource-efficient infrastructures for IDS deployment. Although explicit deployment and maintenance cost metrics were not collected in this study, the observed reductions in storage requirements and improvements in resource utilization suggest the potential for lowering deployment and maintenance costs.

Experimental results demonstrate that the containerized NFV-IDS requires approximately 90% less storage compared to VM-based implementations, highlighting its efficiency in resource utilization.

Performance analysis in a web server environment further confirms that the NFV-IDS can effectively share host resources. However, it is necessary to limit the consumption of available resources.

Among the evaluated scenarios, HTTP-based attacks emerged as the primary source of memory bottlenecks. While other attack types can also strain system resources if undetected, HTTP traffic had the most pronounced impact on memory usage in this study.

Overall, findings indicate that although NFV-IDS technology is still maturing, it presents considerable advantages, particularly in optimizing resource usage and supporting flexible, software-defined network infrastructures.

The modular and automated nature of the proposed testbed framework makes it suitable for future extensions. This adaptability ensures that the framework can continue to support research on the interaction between NFV, container-based virtualization, and network security under diverse conditions. Building upon this flexibility, future work will extend the present study along six main directions:

1. Building on our observation that variations in metrics such as CPU, memory, number of threads, and protocol type could serve as indirect signatures for anomaly detection, we plan to investigate how these resource dynamics can be leveraged to improve IDS design;
2. We will explore optimization techniques to overcome virtualization overhead in NFV-based IDS deployments, including hardware acceleration, optimized data plane technologies, host resource tuning strategies, and microservice-based designs to improve scalability and resource efficiency;
3. We intend to integrate AI-driven IDS solutions to enhance adaptability against evolving DDoS attack patterns and to evaluate NFV-IDS deployments in edge environments, with an emphasis on performance efficiency and minimizing energy consumption;
4. We acknowledge that the experiments were conducted with only three attackers due to resource constraints. While sufficient to analyze differences in resource consumption, this setup does not fully capture large-scale DDoS behavior. Expanding the number of attackers and attack types, therefore, represents an important direction for future work;
5. We also intend to include a detailed quantitative evaluation of deployment and maintenance costs, in order to complement the qualitative assessment presented in this study;
6. Finally, future work will also extend the evaluation beyond DoS attacks to include other categories of intrusions, such as probing, privilege escalation, and malware injection, in order to provide a more comprehensive assessment of IDS performance.

Declarations

Acknowledgements

The authors acknowledge the support from the Graduate Program in Applied Computing (PPGCA) at the Federal University of Technology – Paraná (UTFPR).

Funding

This research did not receive any funding.

Authors' Contributions

All authors were involved in every stage of the work and contributed equally to its completion.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

The source code can be made available by contacting the authors.

References

Abdulganiyu, O., Tchakoucht, T., and Saheed, Y. (2024). Towards an efficient model for network intrusion detection system (IDS): systematic literature review. *Wireless Networks*, 30:453–482. DOI: 10.1007/s11276-023-03495-2.

Adamuz-Hinojosa, O., Ordonez-Lucena, J., Ameigeiras, P., Ramos-Munoz, J. J., Lopez, D., and Folgueira, J. (2018). Automated Network Service Scaling in

NFV: Concepts, Mechanisms and Scaling Workflow. *IEEE Communications Magazine*, 56(7):162–169. DOI: 10.1109/MCOM.2018.1701336.

Albin, E. and Rowe, N. C. (2012). A Realistic Experimental Comparison of the Suricata and Snort Intrusion-Detection Systems. In *International Conference on Advanced Information Networking and Applications Workshops*, pages 122–127. DOI: 10.1109/waina.2012.29.

Ashoor, A. S. and Gore, S. (2012). Intrusion detection system (IDS) & intrusion prevention system (IPS): Case study. *Internatioanl Journal of Scientific & Engineering Research*, 2. Available at: `https://api.semanticscholar.org/CorpusID:110831891`.

AT&T (2019). Beginner's guide: Open source intrusion detection tools. Available at: `https://www.abscomm.net/wp-content/uploads/2020/01/open-source-intrusion-detection.pdf` Accessed: May 2025.

BOJOVIC, Z. (2024). *Application of Network Function Virtualization in Modern Computer Environments*. now Publishers Inc, United States. DOI: 10.1561/9781638283591.

Brumen, B. and Legvart, J. (2016). Performance analysis of two open source intrusion detection systems. In *International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1387–1392. IEEE. DOI: 10.1109/mipro.2016.7522356.

Colella, A. and Colombini, C. M. (2014). Amplification DDoS attacks: Emerging threats and defense strategies. In *International Conference on Availability, Reliability, and Security*, pages 298–310. Springer. Available at: `https://inria.hal.science/hal-01404010v1/document`.

Cresswell, N. and Lapenna, A. (2017). Portainer. Available at: `https://docs.portainer.io/` Accessed: May 2025.

DataReportal, Meltwater, and Social, W. A. (2025). Digital 2025 april global statshot report. Available at: `https://datareportal.com/reports/digital-2025-april-global-statshot` Accessed: May 2025.

ETSI (2019). Network functions virtualisation (NFV). Available at: `https://www.etsi.org/technologies/nfv` Accessed: May 2025.

ETSI, G. N. . (2014). Network Functions Virtualisation (NFV); Architectural Framework. Available at: `https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf` Accessed: May 2025.

ETSI, N. W. p. . (2016). Network operator perspectives on industry progress. Available at: `https://portal.etsi.org/NFV/NFV_White_Paper2.pdf` Acessed: May 2025.

Fadhlillah, A., Karna, N., and Irawan, A. (2021). IDS Performance Analysis using Anomaly-based Detection Method for DOS Attack. In *IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)*, pages 18–22. DOI: 10.1109/IoTaIS50849.2021.9359719.

Fei, X., Liu, F., Zhang, Q., Jin, H., and Hu, H. (2020). Paving the way for nfv acceleration: A taxonomy, survey and future directions. *ACM Computing Surveys (CSUR)*, 53(4):1–42. DOI: 10.1145/3397022.

Forum, W. E. (2025). Global cybersecurity outlook 2025. Available at: `https://reports.weforum.org/docs/WEF_Global_Cybersecurity_Outlook_2025.pdf` Accessed: May 2025.

Gebert, S., Müssig, A., Lange, S., Zinner, T., Gray, N., and Tran-Gia, P. (2017). Processing time comparison of a hardware-based firewall and its virtualized counterpart. In *Mobile Networks and Management*, pages 220–228, Cham. Springer International Publishing. DOI: 10.1007/978-3-319-52712-3$_1$6.

Han, B., Gopalakrishnan, V., Ji, L., and Lee, S. (2015). Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97. DOI: 10.1109/mcom.2015.7045396.

Imperva (2014). What is LOIC - Low Orbit Ion Cannon. Available at: `https://www.imperva.com/learn/ddos/low-orbit-ion-cannon/` Accessed: May 2025.

Inc., D. (2013). Get docker. Available at: `https://docs.docker.com/get-docker/` Accessed: May 2025.

Julienne, T. (2016). SYN Flood Mitigation with SYNsanity. Available at: `https://github.blog/engineering/platform-security/syn-flood-mitigation-with-synsanity/` Accessed: May 2025.

Kuhrer, M., Hupperich, T., Rossow, C., and Holz, T. (2014). Exit from hell? reducing the impact of amplification DDoS attacks. In *USENIX Security Symposium 14*, pages 111–125. Available at: `https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-kuhrer.pdf` Accessed: May 2025.

Kurek, T., Niemiec, M., and Lason, A. (2024). Performance evaluation of a firewall service based on virtualized includeos unikernels. *Scientific Reports*, 14(1). DOI: 10.1038/s41598-024-51167-8.

Mauricio, L. and Rubinstein, M. (2023). A network function virtualization architecture for automatic and efficient detection and mitigation against web application malware. *Journal of Internet Services and Applications*, 14(1):10–20. DOI: 10.5753/jisa.2023.2847.

Merkel, D. (2014). Docker: Lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239):2. Available at: `https://dl.acm.org/doi/10.5555/2600239.2600241`.

Mijumbi, R., Serrat, J., Gorricho, J., Bouten, N., De Turck, F., and Boutaba, R. (2016). Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys Tutorials*, 18(1):236–262. DOI: 10.1109/comst.2015.2477041.

Oliveira, S., Linhares, C., Travençolo, B., and Miani, R. (2020). Investigation of amplification-based DDoS attacks on IoT devices. *INFOCOMP Journal of Computer Science*, 19(1). Available at: `https://infocomp.dcc.ufla.br/index.php/infocomp/article/view/765`.

Rangisetti, A. K. (2024a). *Experiment with VNFs over Docker Containers*, pages 233–295. Apress, Berkeley, CA. DOI: 10.1007/979-8-8688-0497-7_5.

Rangisetti, A. K. (2024b). *Virtualizing Network Functions in Cloud and Telecom Core Networks*, pages 191–231.

Apress, Berkeley, CA. DOI: 10.1007/979-8-8688-0497-7_4.

Roesch, M. (1998). Snort. Available at: https://docs.snort.org/ Accessed: May 2025.

Shayegan, M. J. and Damghanian, A. (2024). A Method for DDoS Attacks Prevention Using SDN and NFV. *IEEE Access*, 12:108176–108184. DOI: 10.1109/ACCESS.2024.3438538.

Sherry, J., Hasan, S., Scott, C., Krishnamurthy, A., Ratnasamy, S., and Sekar, V. (2012). Making middleboxes someone else's problem: network processing as a cloud service. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '12, page 13–24, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2342356.2342359.

Sieklik, B., Macfarlane, R., and Buchanan, W. J. (2016). Evaluation of tftp DDoS amplification attack. *Computers & Security*, 57:67–92. DOI: 10.1016/j.cose.2015.09.006.

Snort Project (2020). Snort users manual. Available at: http://manual-snort-org.s3-website-us-east-1.amazonaws.com/ Accessed: May, 2025.

Susnjara, S. and Smalley, I. (2024). What is docker? Available at: https://www.ibm.com/think/topics/docker.

Tikhe, G. and Patheja, P. (2023). *Mitigation of Distributed Denial of Service (DDoS) Attack Using Network Function Virtualization (NFV)—A Survey*, pages 311–317. DOI: 10.1007/978-981-99-3569-7_22.

Tiwari, M., Kumar, R., Bharti, A., and Kishan, J. (2017). Intrusion detection system. *International Journal of Technical Research and Applications*, 5:2320–8163. Available at: https://www.researchgate.net/publication/316599266_INTRUSION_DETECTION_SYSTEM Accessed: May, 2025.

Tripathi, N., Hubballi, N., and Singh, Y. (2016). How Secure are Web Servers? An Empirical Study of Slow HTTP DoS Attacks and Detection. In *International Conference on Availability, Reliability and Security (ARES)*, pages 454–463. DOI: 10.1109/ARES.2016.20.

Upadhyay, D., Gupta, M., Sharma, K. B., and Upadhyay, A. (2024). Enhancing Network Function Virtualization and Service Function Chaining: Innovative Optimization Strategies and Their Impact. In *International Conference on Pioneering Developments in Computer Science Digital Technologies (IC2SDT)*, pages 153–157. DOI: 10.1109/IC2SDT62152.2024.10696153.

van Cleeff, A., Pieters, W., and Wieringa, R. J. (2009). Security implications of virtualization: A literature study. In *International Conference on Computational Science and Engineering*, volume 3, pages 353–358. DOI: 10.1109/cse.2009.267.

VMware (2005). What is a virtual machine. Available at: https://www.vmware.com/topics/virtual-machine Accessed: May 2025.

Wang, J. (2023). Performant, scalable, and efficient deployment of network function virtualization.

Wang, J., Lévai, T., Li, Z., Vieira, M. A. M., Govindan, R., and Raghavan, B. (2022). Quadrant: a cloud-deployable NF virtualization platform. In *Symposium on Cloud Computing*, SoCC '22, page 493–509, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3542929.3563471.

Yoachimik, O. and Pacheco, J. (2025). Targeted by 20.5 million DDoS attacks, up 358% year-over-year: Cloudflare's 2025 Q1 DDoS Threat Report. Available at: https://blog.cloudflare.com/ddos-threat-report-for-2025-q1/ Accessed: May 2025.

Yusof, M. A. M., Ali, F. H. M., and Darus, M. Y. (2017). Detection and defense algorithms of different types of DDoS attacks. *International Journal of Engineering and Technology*, 9(5):410. Available at: https://link.springer.com/chapter/10.1007/978-981-10-8276-4_35.

Zahoor, S., Ahmad, I., Rehman, A. U., Eldin, E. T., Ghamry, N. A., and Shafiq, M. (2023). Performance Evaluation of Virtualization Methodologies to Facilitate NFV Deployment. *Computers, Materials and Continua*, 75(1):311–329. DOI: 10.32604/cmc.2023.035960.

Çetin, A., Gültekin, D., and and, N. Y. (2025). Implications of NFV-SDN technology: An exploratory study of Turkish telecom industry. *Journal of Global Information Technology Management*, 28(2):111–135. DOI: 10.1080/1097198X.2025.2480971.