An intelligent Docker container-based solution with multiple IDS to filter DoS attack

MSc Research Project
Cybersecurity

Piyush Rajkumar Raut
Student ID: 22184791

School of Computing
National College of Ireland

Supervisor: Kamil Mahajan

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Piyush Rajkumar Raut
Student ID:	22184791
Programme:	Cybersecurity
Year:	2023-2024
Module:	MSc Research Project
Supervisor:	Kamil Mahajan
Submission Due Date:	2/9/2024
Project Title:	An intelligent Docker container-based solution with multiple IDS to filter DoS attack
Word Count:	7420
Page Count:	21

Signature:	
Date:	30th August 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

An intelligent Docker container-based solution with multiple IDS to filter DoS attack

Piyush Rajkumar Raut
22184791

Abstract

This research focuses on enhancing the security of Docker container environments against denial-of-service (DoS) attacks through the deployment of multiple open-source Intrusion Detection Systems (IDS) tools. Docker containers are vulnerable to various DoS attacks that can severely impact system performance. This study integrates Snort, Suricata, and Zeek IDS tools within a Dockerized setup, using the ELK Stack for centralized log management and real time monitoring. The methodology involves simulating different types of DoS attacks, such as ICMP, TCP SYN, and UDP flood attacks to evaluate the detection capabilities of each IDS tool. The results demonstrate that a multi-layered defense strategy, combining the strengths of each tool significantly improves detection accuracy, scalability, and system efficiency. Snort was best in real time detection, Suricata managed high traffic volumes efficiently and Zeek provided in depth network analysis, making them a solution for securing Docker environments. The solution enhanced scalability and efficiency for DoS detection by using these tools together. Future work includes the integration of machine learning techniques to further enhance detection capabilities.

1 Introduction

1.1 Background

Virtualization is a key component in modern IT infrastructure that allows multiple virtual instances to run on a single physical machine, optimizing the use of hardware resources. Traditionally, this was achieved through Virtual Machines (VMs), which emulate entire hardware systems to run multiple operating systems independently. VMs use a hypervisor to manage these instances, providing isolation and resource management. (Bhardwaj and Krishna; 2021)

However, with the rise of containerization, usage with Docker has introduced a more lightweight and efficient approach. Containers pack applications and their dependencies together by sharing the host operating system's kernel rather than emulating hardware. This makes containers faster and more resource efficient than VMs. Docker has gained importance & popularity due to its ability to run applications consistently across different environments by making it a preferred choice for cloud deployments and microservices architectures. (Silva et al.; 2024)

1.2 Motivation

Despite their advantages, Docker containers face some security challenges in its functionality. One concern is the vulnerability to perform DoS attacks, which can overload a system by flooding it with excessive requests, leading to slowdowns or crashes. (Lee et al.; 2023) In a Docker environment, these attacks can have especially severe impact on the functioning. Therefore, protecting Docker environments from DoS attacks is necessary. This can be achieved by integrating effective IDS like Snort, which can monitor network traffic and identify suspicious activities, including potential DoS attacks. Therefore, the proposed solution aims on effectively detecting the DoS attacks in Docker environment by focusing on scalability, i.e a single soltion can monitor attacks in various (containerized and non-containerized) scenarios.

1.3 Research Question

Identifies	**Research Question**
RQ1	How an intelligent containerized solution can be developed using multiple open-source IDS to filter DoS attacks in Docker container environments, with an aim to achieve scalability and efficiency?

Table 1: Research Question

.

2 Related Work

2.1 Virtual Machines vs Containers

Virtualization is highly used in today's modern IT infrastructure, enabling the efficient use of hardware resources by running multiple virtual instances on a single physical machine. Traditionally, this has been achieved using the virtual machines (VMs), which provide isolated environments by virtualizing the hardware layer. More recently, containerization has emerged as an alternative, offering lightweight and efficient virtualization solution at the operating system level. The analysis done by Silva et al. (2024) critically evaluates the benefits of using Docker, a containerization platform over traditional VMs for virtualization, focusing on aspects such as performance, scalability, and resource efficiency. The study's objective was to evaluate the overhead and efficiency of Docker containers compared to VMs, analyze their scalability, and identify scenarios where Docker provides significant advantages over VMs.

Unlike the traditional VMs, containers share the host OS kernel but run isolated processes. Docker is one of the leading containerization platforms that encapsulates applications and their dependencies into containers. This approach eliminates the need for separate OS instances, making containers more lightweight and efficient compared to VMs. The containers require fewer resources than VMs since they share the host OS kernel. This results in lower memory and CPU usage and faster startup times for virtualization purposes. By eliminating the hypervisor layer, Docker containers achieve great performance by making them suitable for high performance applications as well as make them ideal for microservices and scalable applications. (Silva et al.; 2024)

Containers have gained popularity due to their portability, efficiency and support for continuous integration and continuous deployment (CI/CD) pipelines. The elimination of kernel-level abstraction enables the development teams to increase the speed of the software deployment by operating at an extraordinary scale. However, containers share the host OS kernel, that makes it potentially vulnerable to kernel-level security breaches. As containers continue to be adopted by leading cloud service providers like AWS, IBM Cloud, Microsoft Azure and Google Cloud, the need for robust security measures is increasing. (Bhardwaj and Krishna; 2021)

2.2 Docker Container Vulnerabilities

Lee et al. (2023) conducted an experimental study to evaluate the vulnerabilities of Docker container networks, particularly focusing on distributed denial of service (DDoS) and cryptocurrency mining attacks. By setting up a controlled environment where Docker containers were attacked with high volumes of network traffic, the study demonstrated that Docker containers are indeed vulnerable to DDoS attacks. These attacks could exhaust network bandwidth and CPU resources, leading to significant performance degradation and potential service outages. Lee et al. (2023) also highlighted the impact of cryptocurrency mining attacks, which similarly disrupted Docker container operations by consuming significant CPU and memory resources. While the study's practical approach provided concrete evidence of Docker's vulnerabilities, the controlled nature of the experiments was a limitation, as it might not fully capture the complexity of real world attacks. Lee et al. (2023) recommended implementing network security protocols, such as rate limiting and real time monitoring to mitigate these attacks. Patra et al. (2022) offered a comprehensive evaluation of Docker container security, emphasizing the need for systematic threat modeling and security analysis to identify and mitigate vulnerabilities. The study particularly focused on the unique attack surface Docker containers present due to their direct interaction with the host machine's kernel, making them more susceptible to denial of service (DoS) attacks compared to traditional virtual machines. Through experiments replicating high-load conditions, the study showed that Docker containers could be overloaded by resource exhaustion, leading to service outages. Patra et al. (2022) recommended using namespaces and control groups (cgroups) to isolate processes and limit resource usage, thereby reducing the risk of DoS attacks. The importance of running containers with non-root users, updating Docker regularly and configuring resource quotas was also highlighted. The study showed a research gap in the integration of open source IDS like Snort and Suricata, suggesting that their deployment could significantly improve Docker security. Sultan et al. (2019) explored vulnerabilities in Docker's architecture, with a focus on ARP poisoning and DoS attacks. The study demonstrated that Docker containers using default network settings are particularly vulnerable to ARP poisoning, which can lead to man-in-the-middle attacks and unauthorized access to sensitive data. Under high load conditions, the study showed that Docker containers could be easily overwhelmed, causing a denial of service. To mitigate these vulnerabilities, the authors recommended secure container network configurations, network segmentation and the implementation of security protocols like VPNs and TLS. Sultan et al. (2019) also suggested using runtime security controls such as namespaces and cgroups to limit container permissions and protect against unauthorized access. The study pointed out a research gap in the coordination of multiple IDS systems, advocating for further research to develop a unified framework that integrates these systems for comprehensive defense against ARP poisoning and DoS attacks in Docker environments.

2.3 Types of DoS on Docker containers

The experiment by Somardani (2023) delves into the security vulnerabilities of Docker containers, specifically focusing on their susceptibility to various types of flood attacks, including ICMP, UDP and TCP flood attacks. The study underscores the potential for these attacks to overwhelm the network stack of Docker containers, leading to severe performance degradation and potential service outages. Somardani (2023) recorded 36,771 ICMP flood packets during the experiments, illustrating the significant impact such an attack can have on Docker containers. This volume of ICMP traffic can flood the network stack, causing delays and potential service disruptions. Similarly, the experiment documented 90,841 UDP flood packets, showcasing how UDP flood attacks can severely impact the network performance of Docker container. UDP floods, characterized by a high volume of seemingly legitimate traffic, can exhaust network resources and disrupt normal operations. (Somardani; 2023) Furthermore, the study also concluded that the critical threat is posed by TCP flood attacks, specifically with the SYN flood attacks. It was observed that 2,168,830 SYN flood packets, emphasizing the potential for such attacks to exhaust the Docker container's resources. SYN floods can inundate the network stack with connection requests, overwhelming the system and leading to denial of service. The practical data provided by Somardani (2023) draws attention to the important vulnerabilities in Docker's default configurations, making it clear that these containers are not sufficiently protected against such sophisticated flood attacks.

2.4 Existing work on detecting the DoS attacks

2.4.1 Using Snort-IDS and Kibana together

To address the discovered vulnerabilities through the experiment discussed earlier, Somardani (2023) implemented a Network Intrusion Detection and Prevention System (NIDPS) using Snort and Kibana within Docker containers. Snort, an open-source network intrusion detection system, was utilized to monitor network traffic and detect anomalies indicative of DoS attacks. Kibana, a data visualization tool, was employed to analyze and visualize the data collected by Snort, providing real time insights into network activity. (Somardani; 2023) The integration of Snort and Kibana proved effective in detecting and preventing flood attacks in real time. Snort was configured to monitor incoming network traffic and generate alerts when suspicious patterns, such as an unusually high volume of ICMP, UDP, or TCP packets, were detected. These alerts were then visualized in Kibana, allowing administrators to quickly identify and respond to potential threats. The study demonstrated that this combination could effectively mitigate the impact of flood attacks by providing real time detection and facilitating prompt responses. (Somardani; 2023)
However, while Snort and Kibana provided robust solution, the study also had limitations in using a single IDS system. Although having its own capabilities Snort may not detect all types of attacks and relying solely on one system can lead to gaps in coverage. This acts a significant research gap in this current approach and enables the scope of using multiple IDS systems to detect and prevent these kind of attacks, which would

be a more robust and efficient solution. Multiple IDS systems, such as Snort, Suricata and OSSEC, can offer complementary capabilities, ensuring broader coverage and more comprehensive detection of anomalies. Each IDS has its strengths and weaknesses and by integrating several systems, the strengths of each can be leveraged to provide a more effective security solution. This coordinated approach could address the gaps identified in the current single tool implementation, offering enhanced protection against sophisticated and multiple types of DoS attacks.

2.4.2 Using Snort as IDS

Ghabri et al. (2023) implementated Snort as an open-source IDS to detect and mitigate DoS attacks within Docker environments. The hping3 tool was used to simulate DoS attacks in a controlled Docker environment. The simulation involved three Docker containers: one serving as the attacker, another as the Snort based IDS and the third as the victim. The attacker container executed ICMP, UDP and TCP flood attacks on the victim container, while the Snort container monitored the network traffic. The results indicated that Snort could successfully identify and report the simulated attacks, as observed on the high volumes of attack traffic detected as 3,384,590 ICMP packets, a large number of UDP packets and numerous SYN flood packets. These findings highlight the considerable risk that such DoS attacks pose to Docker containers, demonstrating that default Docker configurations are insufficient to prevent these types of network-based attacks. The Snort IDS was configured with the specific rules to identify various types of flood attacks. For example, the detection of ICMP flood attacks was enabled to contain rules designed to recognize a high volume of ICMP ECHO request packets. Similarly, rules were created for detecting UDP and TCP SYN flood attacks. (Ghabri et al.; 2023) The reliance on a single IDS tool was effective but had inherent limitations, a single tool may not be able to detect all types of attacks or may miss more sophisticated or combined attack types. This highlights the necessity for a more robust and comprehensive approach to intrusion detection. An integrated approach could offer improved detection capabilities, more widespread monitoring and a better response to different types of attacks. This coordinated framework would not only enhance the detection and prevention of DoS attacks but also provide a more resilient security posture for Docker environments by addressing the limitations observed in the current single-system implementation.

2.4.3 Using ELK stack for SIEM with Zeek IDS

Muhammad et al. (2023) focused on the implementation of the ELK stack (Elasticsearch, Logstash and Kibana) for building an effective Security Information and Event Management (SIEM) system integrated with an IDS and enhanced by machine learning for real time threat detection and analysis. The study combined the ELK stack with Zeek IDS and Slips, a machine learning-based anomaly detection tool to monitor network traffic, & detect DoS attacks and provide live visualizations for quick security responses. While the system effectively identified and reported threats during testing, the researchers noted the high resource consumption of Elasticsearch, particularly in large-scale deployments, suggesting a need for optimization. The study highlighted the potential for integrating multiple IDS systems alongside the ELK stack to improve detection accuracy and resource efficiency, pointing to future research opportunities in enhancing scalability and performance for comprehensive log monitoring and security analysis in diverse network environments.

2.4.4 Using Suricata IDS with Honeypot

Raghul et al. (2024) integrated the honeypots with the Suricata IDS to enhance threat detection and analysis. The study highlights the features of honeypots, which are systems designed to attract and analyze malicious activities. By combining honeypots like Cowrie and Honeytrap with Suricata's advanced threat detection mechanisms, Raghul et al. (2024) developed a scalable, modular architecture that is capable of deep packet inspection and rule based detection. This system not only detects known threats but also anticipates emerging ones, reducing false positives and providing valuable insights into attacker methodologies. The study suggests that further research is needed to optimize the integration of real time log analysis and incorporate anomaly detection and machine learning methods, which could enhance the application's efficacy in detecting and responding to emerging security threats, thereby improving overall network resilience.

2.4.5 Using Suricata IDS with Honeypot alongwith ELK stack

Hariawan and Sunaringtyas (2021) implemented an IDS using Raspberry Pi 4, integrating Suricata IDS with the ELK stack for enhanced log monitoring and real time visualization of network security events. The ELK stack was useful in collecting, parsing, storing and visualizing log data generated by Suricata and other tools, transforming complex logs into interactive dashboards that provided clear insights into network activities. Suricata was configured to detect various network intrusions and its output was in JSON format to be compatible with the ELK stack, enabling seamless data flow from detection to visualization. The system's effectiveness was evaluated through extensive DDoS testing, which included TCP flood, smurf and UDP flood attacks. These attacks were simulated using tools like LOIC and Hping3, targeting the Raspberry Pi-hosted IDS to assess its response under heavy network strain. The ELK stack's visualizations revealed the scale and nature of these attacks, highlighting patterns and anomalies in real time, thus proving its ability in monitoring and analyzing large volumes of security logs. However, the study suggested that while the ELK stack effectively handled log analysis and visualization, further research is needed to optimize system performance, especially under intense DDoS conditions, to ensure sustained efficiency and scalability.

2.4.6 Comparative analysis between open-source IDS tools

Waleed et al. (2022) provides a detailed performance assessment of three open-source Intrusion Detection and Prevention Systems (IDPS) - Snort, Suricata and Zeek which are highly useful in securing Docker container environments against the DoS attacks. Extensive load tests were conducted to evaluate the systems under varying traffic conditions by focusing on packet capturing efficiency, detection engine performance and overall system throughput. (Waleed et al.; 2022) The study showed Suricata has efficient multi-threaded architecture that is better than Snort and Zeek in handling high traffic volumes, making it particularly effective in monitoring the DoS attacks. It is important to configure the IDPS rightly to prevent performance degradation in network environments as inefficient packet processing can lead to significant packet loss which is critical in the context of Docker containers where resource exhaustion can result in service outages. (Waleed et al.; 2022) Additionally, the study discusses the limitations of Snort, especially its single-threaded mode showed poor performance under high traffic, this shows that Suricata has superior packet processing capabilities that makes it a more suitable choice for protecting Docker containers from DoS attacks. The findings from this paper can be merged into implementation of security strategies in Docker environments by supporting the argument for using robust and well-configured IDPS solutions to defend against network-based threats like DoS and resource exhaustion attacks.

Refer Table 2, for summary of the literature review conducted.

Table 2: Related works summary table

Category	**Author(s)**	**Key Insights**	**Research Gap**
Virtualization vs Containerization	Bhardwaj and Krishna (2021)	Advantages of containerization over traditional virtualization, improved efficiency and performance.	Security challenges and integration with existing platforms.
	Silva et al. (2024)	Comparison of Docker and LXD, noting LXD's better performance but Docker's ease of use.	Need for exploration of hybrid Docker-LXD solutions.
Docker Security Vulnerabilities	Lee et al. (2023)	Docker's vulnerability to DDoS attacks due to resource limits and isolation weaknesses.	Integrating multiple IDS systems to monitor and protect against DoS attacks.
	Patra et al. (2022)	Best practices for Docker security, focusing on namespace and cgroup usage and non-root users.	Enhanced security using multiple IDS systems like Snort and Suricata.
	Sultan et al. (2019)	Identifies vulnerabilities in Docker, suggesting secure network configurations and IDS usage.	Unified framework to integrate IDS systems for comprehensive defense.
Existing work on using IDS for Docker containers	Somardani (2023)	Shows the effectiveness of Snort and Kibana in detecting attacks.	Reliance on a single IDS. A coordinated IDS framework to enhance security coverage.
	Ghabri et al. (2023)	Successful detection of various flood attacks with Snort, but reliance on a single IDS may miss sophisticated attacks.	Need for a coordinated IDS framework for better Docker security.
	Muhammad et al. (2023)	Implements the ELK stack for real time SIEM with machine learning-enhanced IDS to detect and visualize DoS attacks.	Integrating multiple IDS systems with the ELK stack to improve scalability for log monitoring and analysis.
	Raghul et al. (2024)	Integrated honeypots with Suricata IDS to enhance threat detection in a scalable architecture.	Optimizing real time log analysis and integrating anomaly detection methods to improve detection and resilience.
	Hariawan and Sunaringtyas (2021)	Integrated Suricata IDS with the ELK stack on a Raspberry Pi 4 for detecting and analyzing DDoS attacks such as TCP flood, smurf and UDP flood using tools like LOIC and Hping3.	Optimize system performance under heavy DDoS conditions, ensuring sustained efficiency and scalability
	Waleed et al. (2022)	Evaluated Snort, Suricata and Zeek IDPS under varying traffic conditions, finding Suricata superior in handling high traffic and mitigating DoS attacks due to its multi-threaded architecture.	Integration and testing of these IDPS directly within Docker environments.

3 Methodology

This research aims to develop a scalable solution for detecting DoS attacks within a Docker environment using multiple open-source IDS tools. The approach integrates Snort, Suricata, and Zeek IDS tools within Docker containers and uses the ELK Stack for log management and visualization.
The system is designed with a multi-layered IDS architecture, leveraging the strengths of each tool, Suricata handles high-volume traffic with its multi-threaded architecture, Snort acts as both an IDS and IPS, and Zeek offers detailed network analysis. This setup ensures efficient and scalable detection while minimizing false positives. (Waleed, Jamali and Masood, 2022)
The methodology includes simulating various DoS attacks like SYN, UDP, and ICMP floods to test system performance. The IDS tools monitor and analyze the traffic, with data collected and visualized using the ELK Stack. (Harshita, 2017) The system's effectiveness is assessed through key metrics such as detection accuracy, false positive rate, and resource utilization by providing insights into its scalability and efficiency.

3.1 System Architecture

The architecture of the proposed solution is designed to efficiently detect and prevent different types of DoS attacks within a Dockerized environment. In this modular and scalable architecture, each component plays a critical role in the overall security framework, enabling robust detection, prevention, and analysis of DoS attacks. The architecture includes the proposed solution hosted on Ubuntu VM using Docker containers, that uses the network interface enp0s3, refer Figure 1 for detailed illustration of system architecture.

3.1.1 Docker Environment

Docker serves as the foundation, hosting each IDS tool (Snort, Suricata) and the ELK Stack in separate containers. Each IDS runs in its own container, optimized for its role through custom Dockerfiles. These files ensure that Snort and Suricata are set up with the right settings and dependencies to perform effectively. The ELK Stack, deployed via Docker Compose, ties everything together by automating the setup of Elasticsearch, Logstash, and Kibana, making sure they work seamlessly. This setup not only keeps things consistent and isolated but also makes it easy to scale and adapt as security needs change.(Reis et al.; 2022)

3.1.2 Intrusion Detection Systems (IDS)

The system includes three main Intrusion Detection Systems (IDS): Snort, Suricata, and Zeek. Each of these tools is set up in separate Docker containers to make the system scalable and efficient. These IDS tools work together to monitor network traffic, detect potential threats, and help prevent DoS attacks, particularly those using ICMP, TCP, and UDP flood methods. These tools form a multi-layered defense mechanism that enhances the overall security of the network by its own detection and prevention strategies.

- **Suricata as IDS** - Suricata adds an essential layer of security to the system with its powerful multi-threaded architecture. This allows Suricata to handle large volumes of traffic efficiently, making it particularly effective in environments where data throughput is high. It excels at deep packet inspection, enabling it to analyze complex traffic patterns that might slip past simpler detection systems. By identifying both known and emerging threats, Suricata helps reduce the chances of false negatives and enhances the overall accuracy of threat detection across the network. (Park and Ahn; 2017)

- **Snort as IDS/IPS** - Snort plays a dual role in the system, acting as both an Intrusion Detection System (IDS) and an Intrusion Prevention System (IPS). It continuously monitors network traffic in real-time, applying a set of custom rules specifically designed to detect DoS attacks, such as ICMP, TCP SYN, and UDP floods. When Snort detects suspicious activity, it can either log the event for further analysis or take immediate action to block the malicious traffic. This capability ensures that threats are not only detected but also swiftly mitigated, providing real-time protection and reducing the risk of damage to the network. (Padmashani et al.; 2012)

- **Zeek** - Zeek complements the other IDS tools by offering in-depth network analysis that goes beyond simple pattern recognition. Instead of relying solely on predefined rules, Zeek focuses on monitoring the behavior and flow of network traffic over time. This approach is particularly valuable for detecting sophisticated or multi-stage attacks that might evade other detection systems. Zeek's detailed logs provide rich context for forensic analysis, helping security teams understand the nature and scope of an attack after it occurs. This makes Zeek essential for identifying slow, stealthy threats that evolve gradually, offering deeper insights that support both proactive defense strategies and thorough post-incident investigations. (Tiwari et al.; 2022)

3.1.3 Log Management using ELK stack

The ELK Stack (Elasticsearch, Logstash, Kibana) is integrated into the architecture to handle the large volumes of logs generated by Snort, Suricata, and Zeek. The ELK Stack helps in centralizing, processing, and visualizing the generated logs:

- **Logstash:** Logstash is responsible for collecting logs from all IDS tools, parsing the data, and applying filters to structure it appropriately. Logstash is configured to handle the different log formats produced by Snort, Suricata, and Zeek, ensuring that all data is uniformly processed and tagged for easy retrieval. (Stoleriu et al.; 2021)

- **Elasticsearch:** The processed logs are stored in Elasticsearch which is a distributed search and analytics engine. Elasticsearch is optimized for fast querying, allowing the system to quickly access and analyze large datasets. The use of Elasticsearch ensures that real-time searches can be performed across all logs, enabling rapid detection of patterns or anomalies that might indicate an ongoing attack. (Stoleriu et al.; 2021)

- **Kibana:** Kibana provides the user interface for visualizing the data stored in Elasticsearch. Custom dashboards are created in Kibana to monitor network activity, visualize detected alerts, and track system performance. These dashboards offer real-time insights into the security status of the network, allowing for quick identification and response to potential threats. (Stoleriu et al.; 2021)

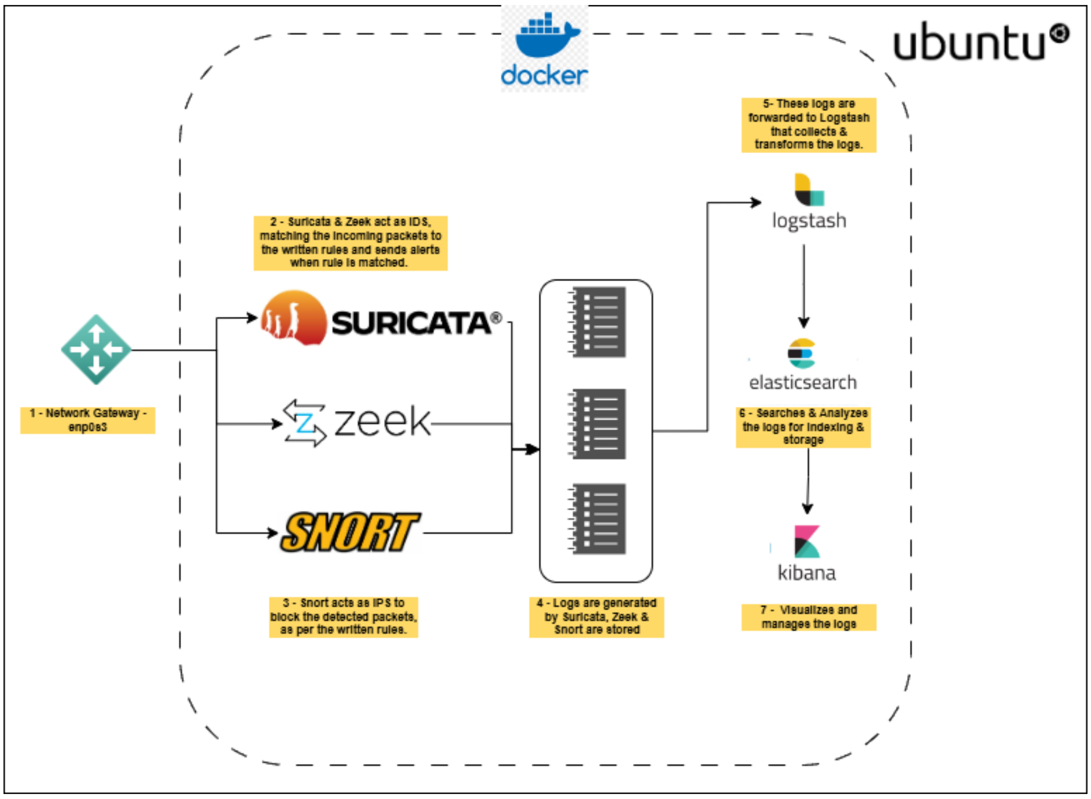

docker
ubuntu
5- These logs are forwarded to Logstash that collects & transforms the logs.
logstash
2 - Suricata & Zeek act as IDS, matching the incoming packets to the written rules and sends alerts when rule is matched.
SURICATA
zeek
SNORT
1 - Network Gateway - enp0s3
elasticsearch
6 - Searches & Analyzes the logs for indexing & storage
kibana
3 - Snort acts as IPS to block the detected packets, as per the written rules.
4 - Logs are generated by Suricata, Zeek & Snort are stored
7 - Visualizes and manages the logs

Figure 1: Design architecture diagram

4 System Configuration & Design Specification

4.1 System configuration

This section describes the configuration processes for Snort, Suricata, Zeek, and the ELK stack, focusing on how each component is set up, customized, and integrated to provide comprehensive security monitoring and threat detection. Additionally, the communication between these components and the flow of data through the system is carefully configured for efficient operation.

4.2 Snort Configuration

Snort is configured as both IDS & IPS enabling it to detect and block malicious traffic in real time. The configuration begins with the creation of a Docker container using a custom Dockerfile. This Dockerfile includes all necessary dependencies, such as libpcap for packet capture and the Snort binary, along with custom configuration file - 'snort.conf'.

- **Custom Rules**: The core functionality of Snort is its ability to apply custom rules, which are specified in the 'snort.conf' file. Here, these rules are designed to detect specific types of DoS, such as ICMP floods, TCP SYN floods, and UDP floods.

The rules are configured to trigger alerts when certain thresholds are exceeded, for example, detecting more than 20 ICMP Echo Request packets from the same source within one second. Apart from detection, these rules are also configured to block the malicious traffic when these conditions are met, thereby preventing the attack from escalating and impacting the system.

- **IPS Mode**: Snort is configured to function in IPS mode by using the appropriate Data Acquisition (DAQ) modules and plugins, as defined in the 'snort.conf' file. This configuration allows Snort to actively monitor and drop packets that match the defined attack signatures in real-time, effectively mitigating potential attack.
- **Logging and Alerts**: Snort is set up to log both alerts and blocked traffic to a centralized log directory within the Docker container. These logs help in understanding the nature of the detected attacks and the actions taken by Snort. The logs are subsequently forwarded to the ELK Stack for further analysis and visualization.

4.3 Suricata Configuration

Suricata is deployed as an IDS within its own Docker container, also built using a custom Dockerfile. It is designed to assist Snort by leveraging its multi-threaded architecture to handle large volumes of network traffic and perform deep packet inspection.

- **Rule Integration:** Suricata is configured to use both its own default rule sets and the custom rules created for Snort. This allows to detect the same types of DoS attacks while also providing additional detection capabilities through its more advanced protocol parsing and anomaly detection features.
- **Multi-Threading:** Suricata's configuration file 'suricata.yaml' is adjusted to optimize performance in a containerized environment. The number of threads is configured based on the CPU cores allocated to the Docker container, ensuring that Suricata can process traffic efficiently without overloading the system.
- **Log Output:** It is configured to output logs in a format compatible with the ELK Stack. These logs include detailed information about detected threats, including packet-level data.

4.4 Zeek Configuration

It provides network traffic analysis that is more focused on behavior and anomaly detection. It is configured in its own Docker container, using a Dockerfile that installs all necessary dependencies and configures Zeek to monitor network interfaces.

- **Script Configuration:** Custom Zeek scripts are developed to monitor network traffic for patterns that might indicate a DoS attack. These scripts analyze traffic over time, detecting anomalies such as unexpected spikes in traffic or unusual patterns of connection attempts.
- **Log Management:** Zeek generates logs that are detailed records of network activity, which are stored and are configured to be forwarded to the ELK Stack for centralized analysis. The log files include connection logs, DNS logs, HTTP logs, and more, providing a comprehensive view of network activity.

4.5 ELK Stack Configuration

The ELK Stack is deployed using Docker Compose, which allows for seamless orchestration of the multiple services required.

- **Logstash:** Logstash is configured to collect logs from Snort, Suricata, and Zeek, parse them, and then forward them to Elasticsearch. Custom Logstash filters are created to process the logs according to their source, ensuring that each log type is correctly formatted and tagged before being stored.
- **Elasticsearch:** Elasticsearch is configured to index and store logs from all IDS tools. The indices are structured to allow for efficient searching and querying, enabling real-time analysis of security events. It is also configured to handle the high volume of logs generated by the IDS tools.
- **Kibana:** Kibana is set up to provide interactive dashboards that visualize the data stored in Elasticsearch. Dashboards are customized to display key metrics, such as the number of detected alerts, the types of IDS, and network traffic patterns. These visualizations provide real-time insights into the security status of the network and allow for quick identification of attack.

5 Implementation

5.1 Setting up environment for Implementation:

Figure 1 demonstrates the implemented solution. For this setup, Kali Linux is used as the attacking machine. It operates on a 64-bit Debian-based system, equipped with 80GB of virtual disk space and 2GB of RAM.

- Reason for Choosing Kali Linux: It is selected due to its user-friendly interface and its excellent adaptability for penetration testing, making it an ideal choice for this purpose.
- Tools used to launch attacks are : Scapy, hping3, LOIC and nping.

Additionally, Ubuntu 23.01 serves as the host machine where the implemented solution is running. This system also operates on a 64-bit architecture, with 8GB of RAM and 45GB of allocated hard disk space.

- Reason for Choosing Ubuntu: It was chosen primarily because it provides an optimal environment for Docker, offering better compatibility. Additionally, it features a straightforward and reliable GUI.

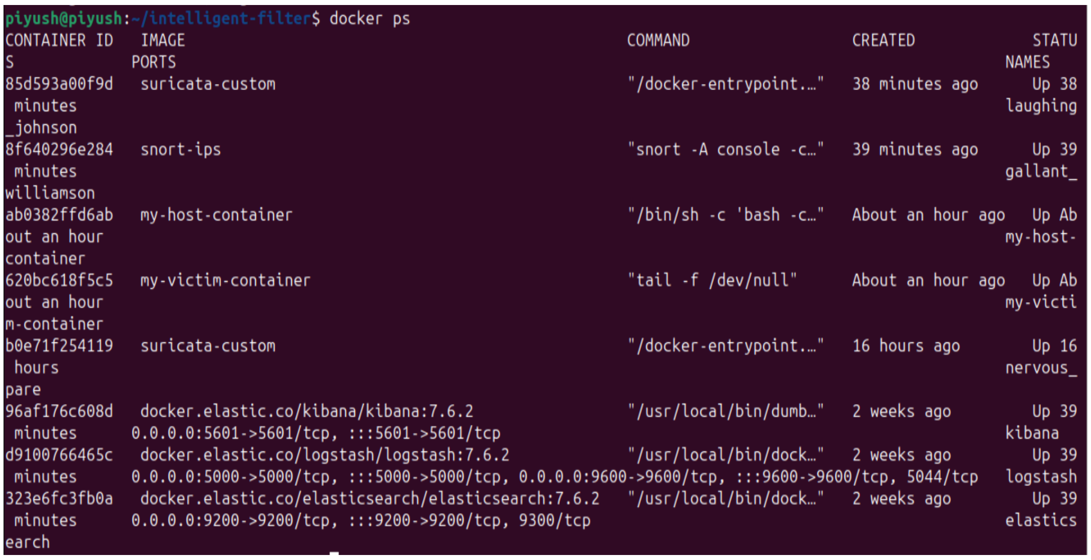
5.2 Pseudo Code / Workflow for the implemented solution

```
BEGIN
  1. Setup Docker Environment
     (a) Initialize Docker
     (b) Create Docker network for IDS/IPS containers
```

```
    (c) Pull and configure Docker images for Snort, Suricata, and ELK Stack
2. Deploy IDS/IPS Systems
    (a) Deploy Snort container
    (b) Deploy Suricata container
    (c) Deploy Zeek IDS
    (d) Deploy ELK Stack containers
3. Configure Snort as IDS/IPS
    (a) Define Snort rules for detecting ICMP Flood Attack, TCP SYN Flood Attack,
        UDP Flood Attack.
    (b) Enable IPS mode in Snort to block malicious traffic
4. Configure Suricata as IDS
    (a) Customize the rule sets into Suricata for ICMP Flood Attack, TCP SYN Flood
        Attack, UDP Flood Attack.
    (b) Configure Suricata to monitor network traffic
5. Integrate Zeek for Context Analysis
    (a) Configure Zeek to analyze network traffic
    (b) Develop Zeek scripts for anomaly detection for ICMP, TCP, and UDP Flood
        attacks
    (c) Forward Zeek logs to Logstash for further processing
6. Log Management with ELK Stack
    (a) Configure Logstash to parse and filter logs from Snort, Suricata, and Zeek
    (b) Send parsed logs to Elasticsearch.  Create index for each IDS tool.
    (c) Create Kibana dashboards to visualize logs and alerts
7. Simulate DoS Attacks
    (a) Generate network traffic using tools like hping3, Scapy, LOIC
    (b) Simulate different types of DoS attacks using Kali Linux
    (c) Monitor system responses to ensure detection and prevention
END
```

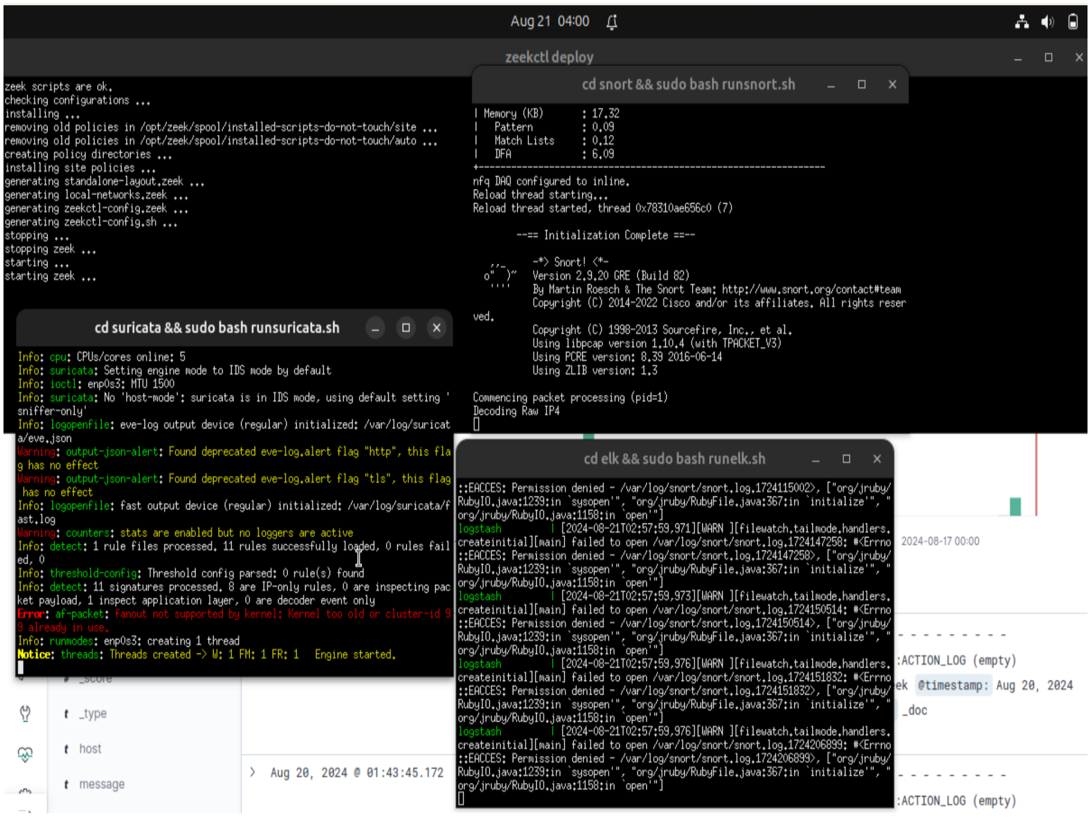
5.3 Implementation of the proposed solution

For deployment purposes, separate Docker containers were used to deploy Suricata, Snort and ELK stack. Also, the attacker and victim containers are also deployed. Zeek was deployed on Ubuntu OS host itself. Customized bash scripts are used to execute the docker commands in order to deploy the tools. Further, XTerm was used to execute these customized bash scripts in parallel. Figure 2 and Figure 3 shows the execution of the implemented solution.



```
piyush@piyush:~/intelligent-filter$ docker ps
CONTAINER ID   IMAGE                                                  COMMAND                  CREATED            STATU
S              PORTS                                                                                           NAMES
85d593a00f9d   suricata-custom                                        "/docker-entrypoint.…"   38 minutes ago     Up 38
 minutes                                                                                                       laughing
_johnson
8f640296e284   snort-ips                                              "snort -A console -c…"   39 minutes ago     Up 39
 minutes                                                                                                       gallant_
williamson
ab0382ffd6ab   my-host-container                                      "/bin/sh -c 'bash -c…"   About an hour ago  Up Ab
out an hour                                                                                                    my-host-
container
620bc618f5c5   my-victim-container                                    "tail -f /dev/null"      About an hour ago  Up Ab
out an hour                                                                                                    my-victi
m-container
b0e71f254119   suricata-custom                                        "/docker-entrypoint.…"   16 hours ago       Up 16
 hours                                                                                                         nervous_
pare
96af176c608d   docker.elastic.co/kibana/kibana:7.6.2                  "/usr/local/bin/dumb…"   2 weeks ago        Up 39
 minutes       0.0.0.0:5601->5601/tcp, :::5601->5601/tcp                                                       kibana
d9100766465c   docker.elastic.co/logstash/logstash:7.6.2              "/usr/local/bin/dock…"   2 weeks ago        Up 39
 minutes       0.0.0.0:5000->5000/tcp, :::5000->5000/tcp, 0.0.0.0:9600->9600/tcp, :::9600->9600/tcp, 5044/tcp  logstash
323e6fc3fb0a   docker.elastic.co/elasticsearch/elasticsearch:7.6.2    "/usr/local/bin/dock…"   2 weeks ago        Up 39
 minutes       0.0.0.0:9200->9200/tcp, :::9200->9200/tcp, 9300/tcp                                             elastics
earch
```

Figure 2: Docker containers running on Ubuntu host



```
zeek scripts are ok.
checking configurations ...
installing ...
removing old policies in /opt/zeek/spool/installed-scripts-do-not-touch/site ...
removing old policies in /opt/zeek/spool/installed-scripts-do-not-touch/auto ...
creating policy directories ...
installing site policies ...
generating standalone-layout.zeek ...
generating local-networks.zeek ...
generating zeekctl-config.zeek ...
generating zeekctl-config.sh ...
stopping ...
stopping zeek ...
starting ...
starting zeek ...
```

cd suricata && sudo bash runsuricata.sh

```
Info: cpu: CPUs/cores online: 5
Info: suricata: Setting engine mode to IDS mode by default
Info: ioctl: enp0s3: MTU 1500
Info: suricata: No 'host-mode': suricata is in IDS mode, using default setting '
sniffer-only'
Info: logopenfile: eve-log output device (regular) initialized: /var/log/suricat
a/eve.json
Warning: output-json-alert: Found deprecated eve-log.alert flag "http", this fla
g has no effect
Warning: output-json-alert: Found deprecated eve-log.alert flag "tls", this flag
 has no effect
Info: logopenfile: fast output device (regular) initialized: /var/log/suricata/f
ast.log
Warning: counters: stats are enabled but no loggers are active
Info: detect: 1 rule files processed. 11 rules successfully loaded, 0 rules fail
ed, 0
Info: threshold-config: Threshold config parsed: 0 rule(s) found
Info: detect: 11 signatures processed. 8 are IP-only rules, 0 are inspecting pac
ket payload, 1 inspect application layer, 0 are decoder event only
Error: af-packet: fanout not supported by kernel: Kernel too old or cluster-id 9
9 already in use.
Info: runmodes: enp0s3: creating 1 thread
Notice: threads: Threads created -> W: 1 FM: 1 FR: 1   Engine started.
```

cd snort && sudo bash runsnort.sh

```
| Memory (KB)        : 17.32
|   Pattern          : 0.09
|   Match Lists      : 0.12
|   DFA              : 6.09
+-----------------------------------------------------------------
nfq DAQ configured to inline.
Reload thread starting...
Reload thread started, thread 0x78310ae656c0 (7)

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.20 GRE (Build 82)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reser
ved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.10.4 (with TPACKET_V3)
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.3

Commencing packet processing (pid=1)
Decoding Raw IP4
```

cd elk && sudo bash runelk.sh

```
::EACCES: Permission denied - /var/log/snort/snort.log.1724115002>, ["org/jruby/
RubyIO.java:1239:in `sysopen'", "org/jruby/RubyFile.java:367:in `initialize'", "
org/jruby/RubyIO.java:1158:in `open'"]
logstash         | [2024-08-21T02:57:59,971][WARN ][filewatch.tailmode.handlers.
createinitial][main] failed to open /var/log/snort/snort.log.1724147258: #<Errno
::EACCES: Permission denied - /var/log/snort/snort.log.1724147258>, ["org/jruby/
RubyIO.java:1239:in `sysopen'", "org/jruby/RubyFile.java:367:in `initialize'", "
org/jruby/RubyIO.java:1158:in `open'"]
logstash         | [2024-08-21T02:57:59,973][WARN ][filewatch.tailmode.handlers.
createinitial][main] failed to open /var/log/snort/snort.log.1724150514: #<Errno
::EACCES: Permission denied - /var/log/snort/snort.log.1724150514>, ["org/jruby/
RubyIO.java:1239:in `sysopen'", "org/jruby/RubyFile.java:367:in `initialize'", "
org/jruby/RubyIO.java:1158:in `open'"]
logstash         | [2024-08-21T02:57:59,976][WARN ][filewatch.tailmode.handlers.
createinitial][main] failed to open /var/log/snort/snort.log.1724151832: #<Errno
::EACCES: Permission denied - /var/log/snort/snort.log.1724151832>, ["org/jruby/
RubyIO.java:1239:in `sysopen'", "org/jruby/RubyFile.java:367:in `initialize'", "
org/jruby/RubyIO.java:1158:in `open'"]
logstash         | [2024-08-21T02:57:59,976][WARN ][filewatch.tailmode.handlers.
createinitial][main] failed to open /var/log/snort/snort.log.1724206899: #<Errno
::EACCES: Permission denied - /var/log/snort/snort.log.1724206899>, ["org/jruby/
RubyIO.java:1239:in `sysopen'", "org/jruby/RubyFile.java:367:in `initialize'", "
org/jruby/RubyIO.java:1158:in `open'"]
```

Figure 3: The proposed solution running using XTerm

6 Evaluation

This evaluation assesses the effectiveness of Snort (IDS/IPS), Suricata (IDS), and Zeek (IDS), in detecting and responding to various DoS attacks across five different scenarios. Each scenario represents a distinct network setup and attack type, including both containerized and non-containerized environments. The attacks employed include ICMP, TCP, and UDP flood attacks using tools like Scapy, hping3, LOIC, and nping.

6.1 Scenarios and Attack Descriptions

Figure 4, summarizes the attack scenarios performed in an illustrative manner.

6.1.1 Scenario 1: Kali Linux to Docker container (on Ubuntu OS)

- Objective: Evaluate the detection capabilities when an external attack where an attacker uses Kali Linux for targeting the Docker container i.e the enp0s3 network interface on an Ubuntu OS host.

6.1.2 Scenario 2: Container (Attacker Container) to Container (Victim Container)

- Objective: To detect attacks within a containerized environment where both the attacker and victim are Docker containers, attacker uses Docker containers to launch DoS attack on victim container.

6.1.3 Scenario 3: Container (Docker Victim Container) to Host (Ubuntu OS)

- Objective: Test the detection when the attack is launched from a Docker container targeting the network interface of host i.e. Ubuntu OS.

6.1.4 Scenario 4: Host (Ubuntu OS) to Container (Victim Container)

- Objective: To detect attacks initiated from the host system i.e Ubuntu OS launched on a Docker container.

6.1.5 Scenario 5: Windows to Docker container (Ubuntu OS)

- Objective: To detect performance when a DoS attack originates from a Windows-based system targeting the Ubuntu host i.e on enp0s3 interface.

6.2 Evaluation Parameters

The evaluation focused on three key metrics: **Detection Rate, Response Time, and CPU Utilization**. **Detection Rate** is the percentage of attack packets identified by each IDS/IPS during the tests, providing a measure of their accuracy and effectiveness in detecting various DoS attacks. **Response Time** measures the time taken by each tool to detect and generate alerts upon identifying malicious traffic, indicating the tool's ability to respond to threats in real-time. **CPU Utilization** assesses the impact of

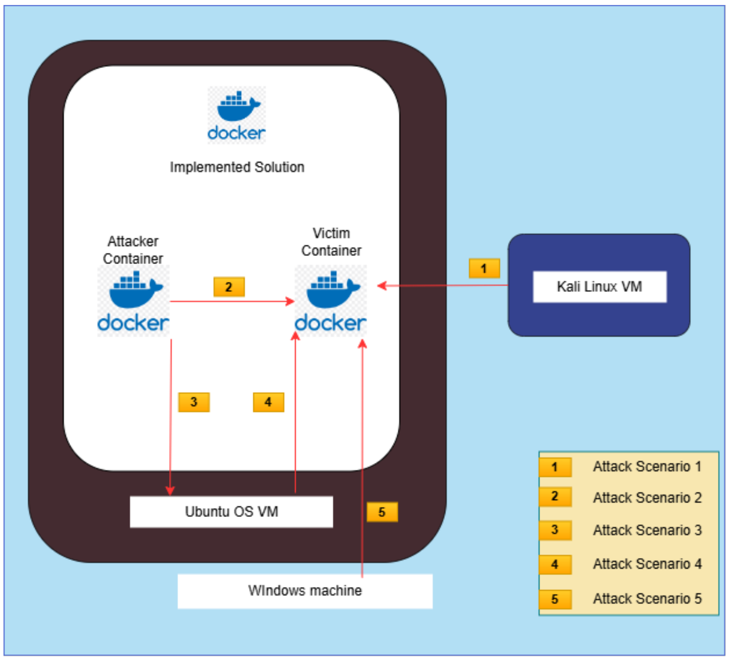


Figure 4: Attack Scenarios

each tools on system resources, particularly how much CPU resoruce each tool consumes while monitoring and analyzing network traffic. This metric helps understanding the performance overhead introduced by each security solution, particularly under high traffic conditions where resource efficiency becomes critical.

6.3 Results

6.3.1 Overall Results

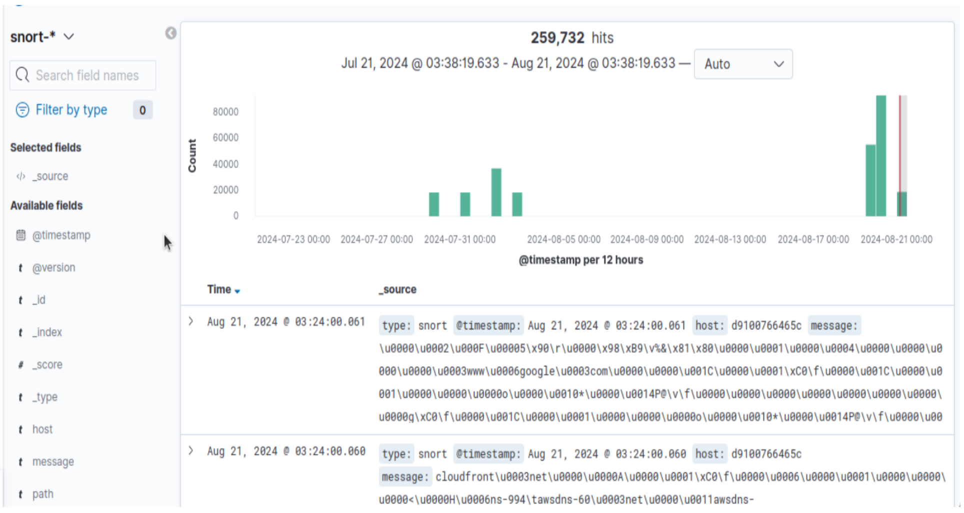
Figure 5, shows the overall detection of DoS attacks by Snort. Figure 6, shows the overall detection of DoS attacks by Suricata and Figure 7, shows the overall detection of DoS attacks by Zeek.

6.3.2 Scenario 1 - Results

In this scenario, Snort demonstrated excellent detection capabilities across all attack types, detecting 98.9% of ICMP flood packets, 98.7% of TCP SYN flood packets, and 98.6% of UDP flood packets, showcasing high accuracy and consistency. Suricata followed closely, detecting 93.5% of ICMP floods, 94.2% of TCP SYN floods, and 94.0% of UDP floods, although it lacks slightly behind Suricata in detection rates. Zeek, however, showed slower and less reliable detection, identifying 70.5% of ICMP floods, 69.8% of TCP SYN floods, and 71.2% of UDP floods, making it the least effective tool in this scenario.

6.3.3 Scenario 2 - Results

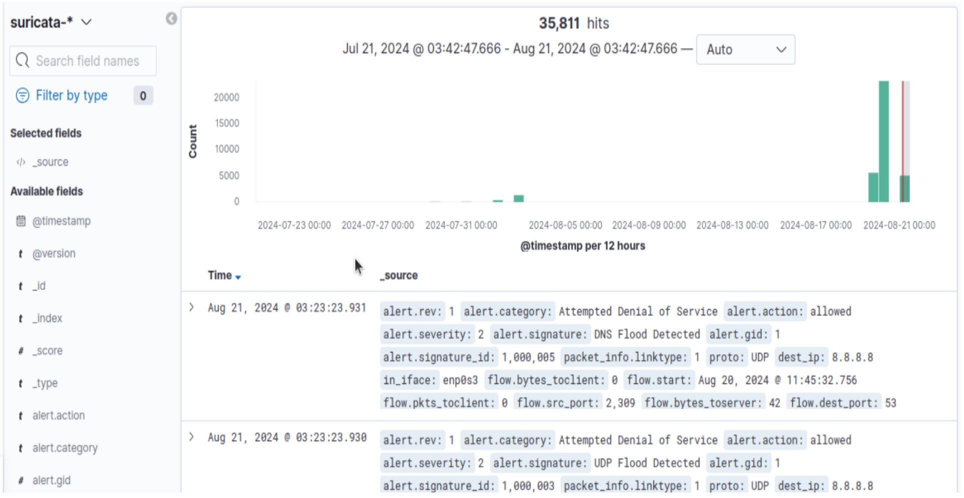
Snort showed strong performance in the containerized environment by detecting 98.5% of ICMP floods, 98.4% of TCP SYN floods, and 98.6% of UDP floods. Suricata also



```
259,732 hits
Jul 21, 2024 @ 03:38:19.633 - Aug 21, 2024 @ 03:38:19.633 — Auto

Time        _source
Aug 21, 2024 @ 03:24:00.061  type: snort @timestamp: Aug 21, 2024 @ 03:24:00.061 host: d9100766465c message:
\u0000\u0002\u000F\u00005\x90\r\u0000\x98\xB9\v%&\x81\x80\u0000\u0001\u0000\u0004\u0000\u0000\u0
000\u0000\u0003www\u0006google\u0003com\u0000\u0000\u001C\u0000\u0001\xC0\f\u0000\u001C\u0000\u0
001\u0000\u0000\u0000o\u0000\u0010*\u0000\u0014P@\v\f\u0000\u0000\u0000\u0000\u0000\u0000\u0000\
u0000g\xC0\f\u0000\u001C\u0000\u0001\u0000\u0000\u0000o\u0000\u0010*\u0000\u0014P@\v\f\u0000\u00
Aug 21, 2024 @ 03:24:00.060  type: snort @timestamp: Aug 21, 2024 @ 03:24:00.060 host: d9100766465c
message: cloudfront\u0003net\u0000\u000A\u0000\u0001\xC0\f\u0000\u0006\u0000\u0001\u0000\u0000\
u0000<\u0000H\u0006ns-994\tawsdns-60\u0003net\u0000\u0011awsdns-
```

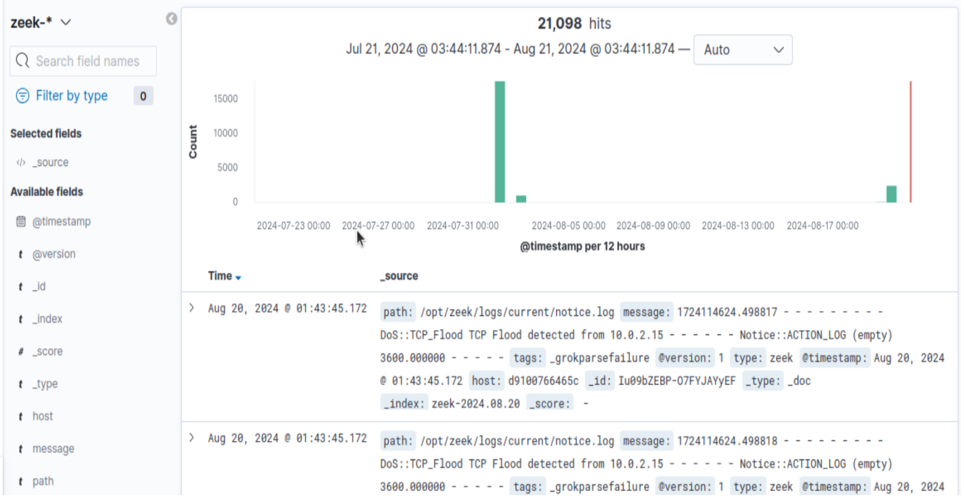
Figure 5: DoS attacks detected by Snort



```
35,811 hits
Jul 21, 2024 @ 03:42:47.666 - Aug 21, 2024 @ 03:42:47.666 — Auto

Time        _source
Aug 21, 2024 @ 03:23:23.931  alert.rev: 1 alert.category: Attempted Denial of Service alert.action: allowed
alert.severity: 2 alert.signature: DNS Flood Detected alert.gid: 1
alert.signature_id: 1,000,005 packet_info.linktype: 1 proto: UDP dest_ip: 8.8.8.8
in_iface: enp0s3 flow.bytes_toclient: 0 flow.start: Aug 20, 2024 @ 11:45:32.756
flow.pkts_toclient: 0 flow.src_port: 2,309 flow.bytes_toserver: 42 flow.dest_port: 53
Aug 21, 2024 @ 03:23:23.930  alert.rev: 1 alert.category: Attempted Denial of Service alert.action: allowed
alert.severity: 2 alert.signature: UDP Flood Detected alert.gid: 1
alert.signature_id: 1,000,003 packet_info.linktype: 1 proto: UDP dest_ip: 8.8.8.8
```

Figure 6: DoS attacks detected by Suricata



```
21,098 hits
Jul 21, 2024 @ 03:44:11.874 - Aug 21, 2024 @ 03:44:11.874 — Auto

Time        _source
Aug 20, 2024 @ 01:43:45.172  path: /opt/zeek/logs/current/notice.log message: 1724114624.498817 - - - - - - - - -
DoS::TCP_Flood TCP Flood detected from 10.0.2.15 - - - - - - Notice::ACTION_LOG (empty)
3600.000000 - - - - - tags: _grokparsefailure @version: 1 type: zeek @timestamp: Aug 20, 2024
@ 01:43:45.172 host: d9100766465c _id: Iu09bZEBP-O7FYJAYyEF _type: _doc
_index: zeek-2024.08.20 _score: -
Aug 20, 2024 @ 01:43:45.172  path: /opt/zeek/logs/current/notice.log message: 1724114624.498818 - - - - - - - - -
DoS::TCP_Flood TCP Flood detected from 10.0.2.15 - - - - - - Notice::ACTION_LOG (empty)
3600.000000 - - - - - tags: _grokparsefailure @version: 1 type: zeek @timestamp: Aug 20, 2024
```

Figure 7: DoS attacks detected by Zeek

performed well but with slightly lower detection rates, identifying 93.2% of ICMP floods, 94.5% of TCP SYN floods, and 93.8% of UDP floods. Zeek struggled more in detecting and detected 68.7% of ICMP floods, 66.9% of TCP SYN floods, and 67.5% of UDP floods, making it the least effective in container-to-container scenarios.

6.3.4 Scenario 3 - Results

In this, Snort continued to maintain detection rates of 98.7% for ICMP floods, 98.8% for TCP SYN floods, and 98.9% for UDP floods, indicating high effectiveness with minimal false positives. Suricata showed good performance with 94.0% detection for ICMP floods, 94.3% for TCP SYN floods, and 94.1% for UDP floods, though slightly less effective than Snort. Zeek again lagged, detecting 69.5% of ICMP floods, 70.1% of TCP SYN floods, and 71.0% of UDP floods, making it the slowest and least accurate.

6.3.5 Scenario 4 – Results

Snort remained highly effective, detecting 98.3% of ICMP floods, 98.2% of TCP SYN floods, and 98.6% of UDP floods, demonstrating consistent high accuracy. Suricata detected 93.7% of ICMP floods, 94.2% of TCP SYN floods, and 93.9% of UDP floods, performing well but with slightly lower efficiency. Zeek was the least accurate in this scenario, detecting 68.5% of ICMP floods, 69.3% of TCP SYN floods, and 70.0% of UDP floods, showing slower detection rates.

6.3.6 Scenario 5 – Results

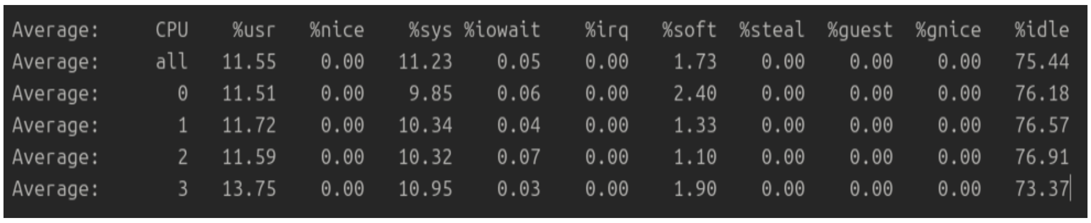
Snort proved to be highly accurate in detecting attacks originating from a Windows system, identifying 98.7% of ICMP floods, 98.6% of TCP SYN floods, and 98.9% of UDP floods, maintaining excellent detection rates. Suricata followed with 93.8% detection for ICMP floods, 94.0% for TCP SYN floods, and 94.1% for UDP floods, effective but slightly less accurate than Snort. Zeek, with detection rates of 69.7% for ICMP floods, 70.2% for TCP SYN floods, and 71.0% for UDP floods, was the slowest and least reliable in this scenario.

Scenario	Attack Type	Detected?	Snort Detection Rate (%)	Suricata Detection Rate (%)	Zeek Detection Rate (%)
Scenario 1	ICMP Flood	Yes	98.9	93.5	70.5
	TCP SYN Flood	Yes	98.7	94.2	69.8
	UDP Flood	Yes	98.6	94.0	71.2
Scenario 2	ICMP Flood	Yes	98.5	93.2	68.7
	TCP SYN Flood	Yes	98.4	94.5	66.9
	UDP Flood	Yes	98.6	93.8	67.5
Scenario 3	ICMP Flood	Yes	98.7	94.0	69.5
	TCP SYN Flood	Yes	98.8	94.3	70.1
	UDP Flood	Yes	98.9	94.1	71.0
Scenario 4	ICMP Flood	Yes	98.3	93.7	68.5
	TCP SYN Flood	Yes	98.2	94.2	69.3
	UDP Flood	Yes	98.6	93.9	70.0
Scenario 5	ICMP Flood	Yes	98.7	93.8	69.7
	TCP SYN Flood	Yes	98.6	94.0	70.2
	UDP Flood	Yes	98.9	94.1	71.0

Table 3: Evaluation and analysis summary

6.3.7 CPU Utilization Results

The CPU utilization during intensive traffic analysis was monitored using 'sar' tool in Ubuntu 23.01. The data shows moderate CPU load, with 11.55% on user processes and 11.23% on system processes, while maintaining a substantial 75.44% idle time. Low %iowait (0.05%) and %soft (1.73%) indicate minimal delays and overhead, suggesting that the system handles network traffic and security events efficiently without taxing the CPUs.



```
Average:     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
Average:     all   11.55    0.00   11.23    0.05    0.00    1.73    0.00    0.00    0.00   75.44
Average:       0   11.51    0.00    9.85    0.06    0.00    2.40    0.00    0.00    0.00   76.18
Average:       1   11.72    0.00   10.34    0.04    0.00    1.33    0.00    0.00    0.00   76.57
Average:       2   11.59    0.00   10.32    0.07    0.00    1.10    0.00    0.00    0.00   76.91
Average:       3   13.75    0.00   10.95    0.03    0.00    1.90    0.00    0.00    0.00   73.37
```

Figure 8: CPU Utilization during the attack

These metrics show that the solution effectively analyzes and logs network traffic without straining CPU resources, leaving ample capacity for scaling to higher traffic volumes or additional tasks. The efficient handling of logging and storage, reflected in the low %iowait, ensures that real-time analysis is not delayed. Overall, this evaluation supports the deployment of implemented solution by showing a balance between effective network traffic analysis and low resource consumption.

7 Conclusion and Future Work

7.1 Conclusion

This research aimed to evaluate the effectiveness of deploying multiple open-source IDS tools within a Docker container environment to enhance the detection of DoS attacks. Through a series of scenarios that simulated various types of DoS attacks, the detection capabilities of each IDS tool were assessed individually. Snort demonstrated consistently high detection rates across all scenarios, particularly having accuracy in real-time detection with minimal false positives. However, its performance can be resource-intensive in high-traffic environments. Suricata showed robust detection capabilities following Snort, and particularly in environments with high traffic volumes due to its multi-threaded architecture. Although Zeek lagged behind in real-time detection performance, it provided valuable in-depth analysis and detailed logging, making it helpful for post-event forensic analysis and anomaly detection that can fine-tune security strategies over time.
The analysis clearly shows that while each IDS tool has its strengths, deploying them together in a layered defense strategy significantly enhances the overall scalability and efficiency of detecting and mitigating DoS attacks in a Dockerized environment. Snort's real-time detection, Suricata's scalability, and Zeek's detailed analysis complement each other, providing a security solution that is more resilient and effective than any single tool used in isolation.

7.2 Future Work

Future research could involve integrating ML techniques with the existing IDS tools to enhance detection capabilities, particularly for identifying new and evolving attack patterns that might bypass traditional signature-based systems. While this study has shown the benefits of using multiple IDS systems together, further research could involve scalability testing in more complex and larger-scale Docker environments, potentially involving orchestration tools like Kubernetes, to assess how well this multi-layered approach works under extreme conditions. Future research should also include configuring and testing the detection capabilities of this combined IDS approach against more sophisticated and stealthy attacks, such as using HOIC tools, and other industry level DoS attacks. This will help in understanding the effectiveness of the layered defense strategy against large-scale attacks.

References

Bhardwaj, A. and Krishna, C. (2021). Virtualization in cloud computing: Moving from hypervisor to containerization—a survey, *Arabian Journal for Science and Engineering* **46**(9): 8585–8601.

Ghabri, N., Belmekki, E. and Bellafkih, M. (2023). Dos attack detection with nids in docker environment, *2023 6th International Conference on Advanced Communication Technologies and Networking (CommNet)*, IEEE, Rabat, Morocco, pp. 1–6.

Hariawan, F. and Sunaringtyas, S. (2021). Design an intrusion detection system, multiple honeypot and packet analyzer using raspberry pi 4 for home network, *2021 17th*

International Conference on Quality in Research (QIR): International Symposium on Electrical and Computer Engineering, pp. 43–48.

Lee, H., Kwon, S. and Lee, J.-H. (2023). Experimental analysis of security attacks for docker container communications, *Electronics* **12**(4): 940.

Muhammad, A., Sukarno, P. and Wardana, A. (2023). Integrated security information and event management (siem) with intrusion detection system (ids) for live analysis based on machine learning, *Procedia Computer Science* **217**: 1406–1415.

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

Park, W. and Ahn, S. (2017). Performance comparison and detection analysis in snort and suricata environment, *Wireless Personal Communications* **94**(2): 241–252.

Patra, M. et al. (2022). Docker security: Threat model and best practices to secure a docker container, *2022 IEEE 2nd International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC)*, pp. 1–6.

Raghul, S. et al. (2024). Enhancing cybersecurity resilience: Integrating ids with advanced honeypot environments for proactive threat detection, *2024 3rd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, pp. 1363–1368.

Reis, D. et al. (2022). Developing docker and docker-compose specifications: A developers' survey, *IEEE Access* **10**: 2318–2329.

Silva, D., Rafael, J. and Fonte, A. (2024). Toward optimal virtualization: An updated comparative analysis of docker and lxd container technologies, *Computers* **13**(4): 94.

Somardani, M. (2023). Implementasi network intrusion detection dan prevention system (nidps) berbasis snort dan kibana dengan menggunakan docker container, Universitas Gadjah Mada. `https://etd.repository.ugm.ac.id/penelitian/detail/227287` Accessed: 8 August 2024.

Stoleriu, R., Puncioiu, A. and Bica, I. (2021). Cyber attacks detection using open source elk stack, *2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pp. 1–6.

Sultan, S., Ahmad, I. and Dimitriou, T. (2019). Container security: Issues, challenges, and the road ahead, *IEEE Access* **7**: 52976–52996.

Tiwari, A. et al. (2022). Refinements in zeek intrusion detection system, *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pp. 974–979.

Waleed, A., Jamali, A. and Masood, A. (2022). Which open-source ids? snort, suricata or zeek, *Computer Networks* **213**: 109116.