Journal of Applied Informatics and Computing (JAIC)
Vol.9, No.3, June 2025, pp. 973~978
e-ISSN: 2548-6861

Analysis of docker container Implementation in SIEM infrastructure

Noper Ardi [1]*, Ahmadi Irmansyah Lubis[2]*, Ikhwan Ash Shafa Arrafi [3]**
* Teknologi Rekayasa Perangkat Lunak, Politeknik Negeri Batam
** Teknologi Rekayasa Multimedia, Politeknik Negeri Batam
noperardi@polibatam.ac.id [1], ahmadi@polibatam.ac.id[2], arrafiikhwan@gmail.com [3]

Article Info

Article history:

Received 2025-04-23
Revised 2025-06-13
Accepted 2025-06-17

Keyword:

Analysis,
Containerization,
Docker,
Wazuh.
Virtualization.

ABSTRACT

It is known that configuring system information and event management (SIEM) infrastructure using conventional virtualization still provides essential functions. However, if a problem occurs such as a configuration error during the staging process or application service failure, the recovery process from the error requires quite a long time. This research aims to explore and analyze the implementation of container technology in the SIEM Infrastructure using the Wazuh platform. The analysis focuses on a Docker-based architecture running Wazuh's core components: the wazuh-indexer, wazuh-manager, and wazuh-dashboard, each in its own container. This approach is evaluated to see how containerization affects SIEM effectiveness and efficiency, particularly in resource utilization and fault recovery. Performance testing carried out on systems using Docker Containers shows lower Memory and CPU usage compared to Conventional Virtualization. The results demonstrate that Docker not only enhances resource efficiency but also improves system resilience, directly impacting SIEM operational functionality.

I. INTRODUCTION

In previous research, wazuh installation was carried out using conventional virtualization in the cloud, which provides essential functions [1]. However, this can be a problem if a configuration error occurs during the staging process. Recovery from these errors can take quite a long time to return the operating system to the state before the error occurred. This can be overcome by utilizing the snapshot feature; however, it can be a concern in terms of costs inasmuch as the VM (Virtual Machine) instances used are on cloud services. Flexibility during migration and maintenance such as upgrades is also a concern when using conventional virtualization. In response to this issue, the container system approach is considered to be an alternative that is worth trying. Therefore, this research aims to explore and analyze the implementation of container technology in the SIEM Infrastructure.

Therefore, using Containers like Docker includes the ability to create images that can be quickly built, shared, and run across multiple environments with flexible configurations. Containers are an approach to application development and management that allows an application and all of its dependencies to be isolated in a container that can be run across multiple environments [2]. Docker containers can optimize resource use more efficiently compared to conventional virtualization.

Docker containers allow users to quickly create and save snapshots of running containers. In the context of Wazuh, this means that the system configuration can be backed up efficiently. If a configuration error occurs, recovery can be performed quickly using the snapshot without needing to restore the entire operating system. This Docker snapshot feature provides great flexibility in error management and reduces associated downtime. Containers differ from virtualization in that they require a separate operating system for each virtual machine, whereas multiple containers can share the same operating system [3]. Additionally, Docker containers can be easily stopped or removed when they are not needed, so users only pay for the resources used while the container is running. This provides better control over operational costs.

As the research conducted by Felani et. al [4], the test results show that optimization making the low specification server can run multiple applications all at once. the Docker container virtualization can tackle the high-cost hardware

requirements. These are proven by resource utilization that relatively low at 15,5% and 18,8% cpu usage percentage respective to its cores and 61.8% of ram usage from 2 GiB total.

Informed by Mulyadi et al [5], regarding the use of the Elastic Stack implementation for a security information management system using the Docker container approach, it shows good performance and low latency. Although the performance with the Docker container approach is not significantly different from the standalone version, resource efficiency can be obtained because using Docker can carry out deployment more than one service instance in one host.

Previous research has focused on deploying Wazuh using conventional virtualization or implementing the Elastic Stack on Docker. However, a direct, systematic performance comparison of a complete Wazuh SIEM stack on Docker versus a traditional VM is not extensively covered. The novelty of this study is therefore explicit: *Unlike previous works that focus on Docker-based IDS or general logging platforms, this study focuses on a comprehensive SIEM stack performance comparison under Docker vs. traditional VM environments, using Wazuh as the subject*. This research provides a quantitative analysis of resource utilization under a simulated attack load, offering clear metrics on the efficiency gains of containerization for a full-featured SIEM solution[6][7][8].

This research provides a comprehensive analysis covering several key operational aspects. The primary focus is on resource efficiency, measured through a direct comparison of CPU and memory utilization between containerized and traditional VM environments. The analysis also extends to deployment and management, where the use of Docker Compose simplifies the definition of multi-container applications. A significant aspect of this study is the evaluation of fault recovery capabilities, highlighting how container snapshots offer a more rapid recovery mechanism compared to traditional VM snapshots. Finally, while this study implements a single-node architecture , it lays the groundwork for evaluating the scalability of a containerized SIEM, as the Wazuh platform itself is designed for multi-node deployments.

II. METHODOLOGY

2.1 Container

Containers are an isolated and lightweight virtualization technique that provides a portable runtime environment without depending on the underlying hardware [6]. Containers are specially encapsulated and secured processes that run on the host system [7]. Virtualization in containers is often called operating system level virtualization, which allows running many applications on one host [4]. Unlike conventional virtualization, a container does not run a full operating system, a container only provides a very minimal operating system environment for running applications that have been packaged in the container, along with the dependencies required by the application in isolation as illustrated in Figure 1.

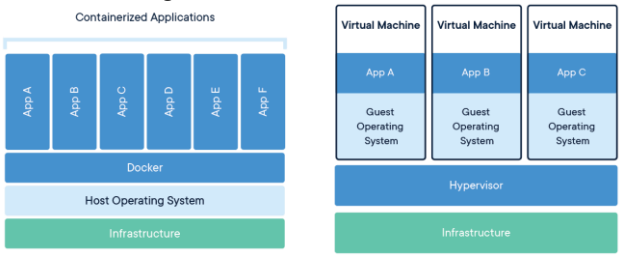

Figure 1. Comparison of Container and Virtual Machine Architectures [2]

Containers are managed by Container managers (or also called engines or daemons), an example of a container manager is Docker. Container-based virtualization such as Docker can be used as a replacement for virtual machines for faster functionality as starting and shutting down containers is much faster.

2.2 Docker

Docker is a platform that provides the ability to create, run, and test applications with a container system. Isolation and security allow users to run multiple containers simultaneously on a particular host. Docker uses resource isolation features of the Linux kernel such as cgroups (Control Groups are Linux kernel features that limit, account for and isolate resource usage) and kernel namespaces (variables or identifiers) to allow Containers to run within a single Linux instance, to avoid overhead [9][10].

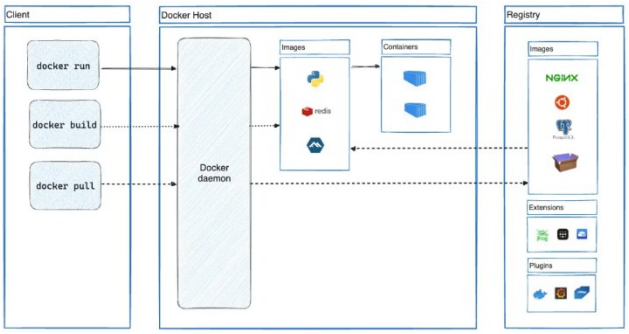

Figure 2. Docker Architecture [1]

Docker has several tools for various deployment methods, one of which is Docker Compose, Docker Compose is a tool that helps define multi-container applications. Using Docker Compose offers several benefits that simplify the development, deployment, and management of containerized applications [9].

2.3 Wazuh

Wazuh is a security platform that provides integrated XDR (Extended detection and response) and SIEM (Security Information and Event Management) services for endpoints. The solution implemented by Wazuh is based on agents

deployed on monitored endpoints, and on three main components: Wazuh Server, Wazuh Indexer, and Wazuh Dashboard [10].

Wazuh deployment can vary depending on the infrastructure architecture used, which in the context of this research will use docker-compose to be multi-container configurable. Wazuh can be deployed as a single-node or multi-node which has a difference in the number of wazuh-indexers deployed and wazuh-managers which are divided into workers and masters. In the context of this research, Wazuh Docker will be deployed as a single-node, namely with each component with one container[11][12][13].

2.4 Proposed system design

A docker host instance will run the docker compose configuration, the docker compose configuration will run three containers, namely wazuh-indexer, wazuh-manager, and wazuh-dashboard. Several client instances are required as agents to send security alert data. The wazuh-manager container will receive security alert data from the agent and process the data and then store it through the wazuh-indexer container. The wazuh-indexer container not only stores data sent by the agent, but also credentials for logging in via the wazuh-dashboard, the wazuh-indexer functions like a database. The private registry is used to store Docker images to speed up the image pull process [14][15][16].

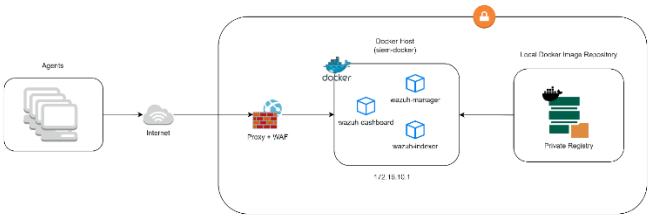

Figure 3. System Topology

2.5 Docker compose system flow

By using the docker-compose up -d command, Docker compose will read the docker-compose.yml configuration file in the directory. If the docker host does not yet have the image required in the docker-compose.yml configuration, Docker will pull the image first[17][18][19]. After the image is finished pulling, Docker compose will continue by building the container according to what has been configured. Figure 3.2 is a system flow diagram in Docker compose.

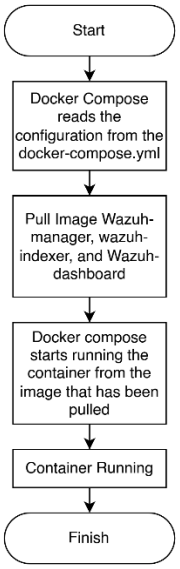

Figure 4. System Flow diagram in Docker Compose

2.6 System specification

The system is run in the Proxmox hypervisor environment with the following specifications:

TABLE 1.
SPECIFICATION OF THE SYSTEM

CPU	(4 vCPU) Intel(R) Xeon(R) CPU E5-2630 v4
RAM	8 GB
DISK	60 GB
OS	Debian 12 “Bookworm”

Table 1 is the specifications that will be used in the system that has been designed as discussed in Sub-chapter 3.1, the virtual machine will later be installed with docker-compose.

2.6 System testing procedure

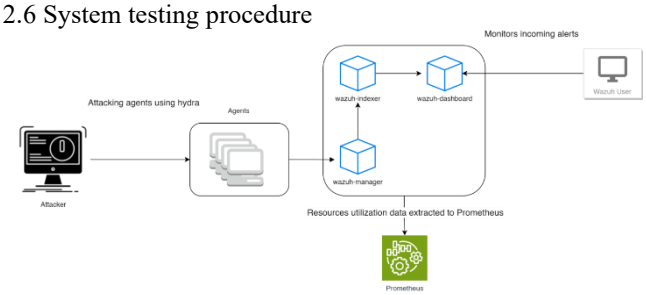

Figure 5. System Testing Procedure Flow

System performance testing has been designed by calculating system performance metrics, namely CPU Utilization and Memory Utilization when idle and when receiving alerts from agents using several testing tools, namely Docker Stats and Prometheus Node-exporter [12]. For comparison, the same test was also carried out on a system using conventional virtualization deployment. The test scenario is carried out by carrying out an attack on the agent, then the agent will send a hit alert to Wazuh. The test

scenario is carried out by carrying out an attack on the agent and then the agent will send a hit alert to Wazuh which is carried out in the form of a bruteforce ssh password attack to agents registered with Wazuh using hydra [20][21].

System performance testing was designed to compare the containerized SIEM deployment against a conventional virtualization setup under both idle and high-load conditions. The SIEM platform used for this research is Wazuh[17][18]. The testing procedure is as follows:

a) Environment Setup: Two identical environments were prepared on a Proxmox hypervisor, each with 4 vCPUs, 8 GB RAM, and a 60 GB disk on Debian 12. One environment runs the Wazuh stack using Docker Compose, while the other runs it on a conventional virtual machine.
b) High-Load Scenario Generation: To simulate a high-load condition, a brute-force SSH password attack was launched against four registered Wazuh agents simultaneously. This attack was executed using the Hydra tool for a duration of 15 minutes, causing the agents to generate and send a continuous stream of security alerts to the Wazuh manager.
c) Performance Measurement: During the 15-minute attack scenario, system performance metrics, namely CPU Utilization and Memory Utilization, were monitored and collected. The primary tools used for measurement were Prometheus Node-exporter for data collection and Grafana for visualization and analysis. Docker Stats was also used for supplementary monitoring of the containerized environment.
d) Comparative Analysis: The collected data from both the Docker and conventional virtualization systems were then compared head-to-head to analyze performance differences under load

III. RESULT AND DISCUSSION

Performance testing on systems that use containers (docker hosts) is carried out by attacking agents registered with wazuh so that the agents will send hit alerts. The registered agents are four agents and send hit alerts at once. Performance monitoring was carried out using Prometheus and then visualized using Grafana.

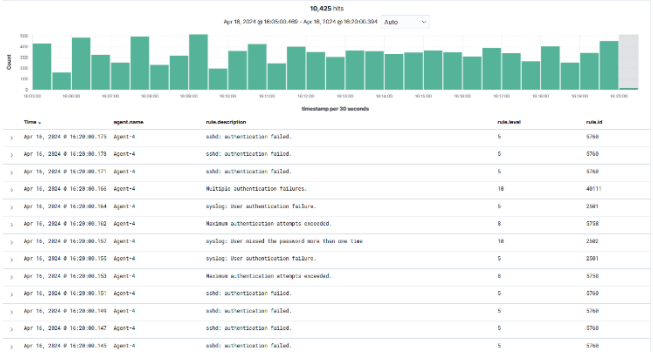

Figure 6. Wazuh receives a hit alert from the agent

As shown in Figure 6, to be able to test the performance of Wazuh when receiving an alert from an agent, an attack was carried out in the form of a bruteforce SSH password to the agent registered with Wazuh using Hydra, the attack was carried out on 4 agents. This test is carried out within a period of 15 minutes.

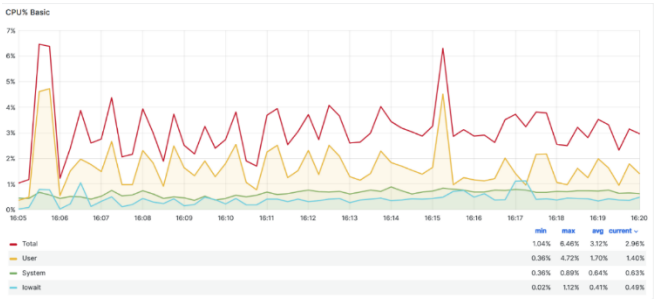

Figure 7. CPU Usage on Docker host when 4 registered agents send alerts

In Figure 7, by testing the attacking agent, the agent will send a hit alert in the time period from 16:05 to 16:20. It was found that CPU usage on the Docker Host increased by a maximum of 6.46% with an average of 3.12%.

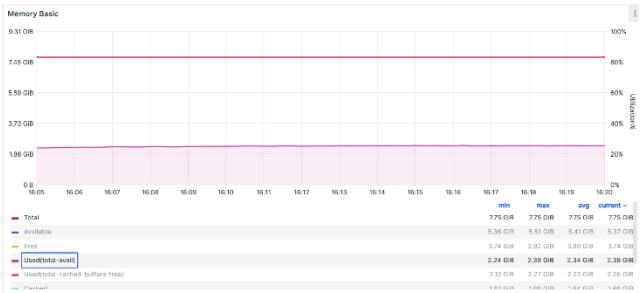

Figure 8. Graph of Memory Usage on Docker Host when 4 agents send alerts

Figure 8 shows that the Docker host's memory usage when receiving alerts has increased with a maximum of 2.39 GiB, a minimum of 2.24 GiB, with an average of 2.34 GiB.

Performance testing on instances running wazuh with Conventional Virtualization is carried out by attacking agents registered with wazuh so that the agent will send a hit alert. The registered agents are four agents and send hit alerts at once. Performance monitoring was carried out using Prometheus and then visualized using Grafana.

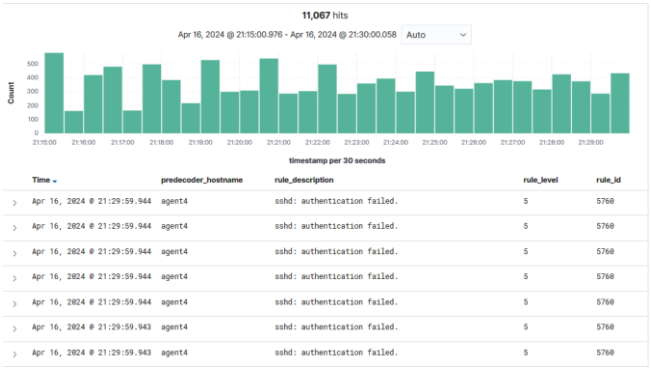

Figure 9. Wazuh in conventional virtualization receives a hit alert from the Agent

As shown in Figure 9, the test was carried out by carrying out an attack in the form of a bruteforce SSH password to a registered agent within a time span of 15 minutes.

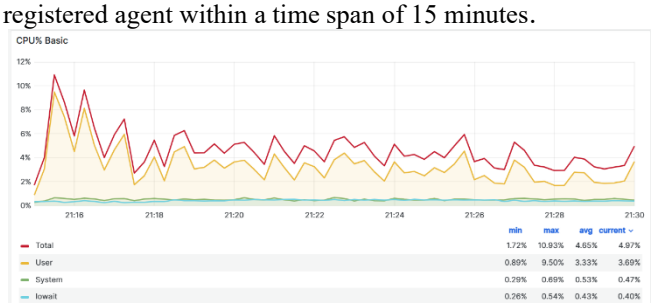

Figure 10. CPU Usage on a conventional virtualization instance when receiving an alert

In Figure 10, by testing the attacking agent, the agent will send a hit alert within 15 minutes. It is found that CPU usage on VM instances has increased by a maximum of 10.93% with an average of 8.58% and a minimum of 1.72%.

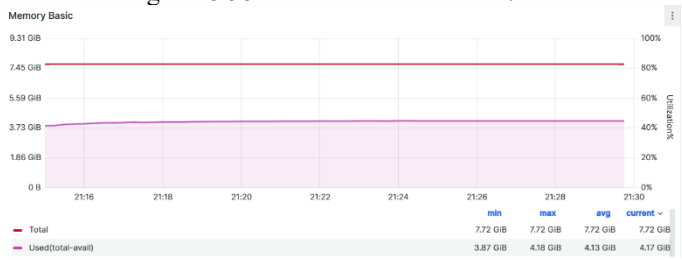

Figure 11. Memory usage on conventional virtualization instances when receiving alerts

In Figure 11, memory usage when testing was carried out increased to a maximum of 4.18 GiB with an average of 4.13 GiB and a minimum of 3.87 GiB.

Analysis of CPU and memory utilization performance testing on both systems using data obtained from previous tests. Data that has been obtained from previous tests is combined and then compared head-to-head, with the blue bar (Docker) being the result of testing on a system that uses containers and the orange bar (Native Virt) being a system that uses conventional virtualization. The graph presented below shows usage data at intervals of every 1 minute.

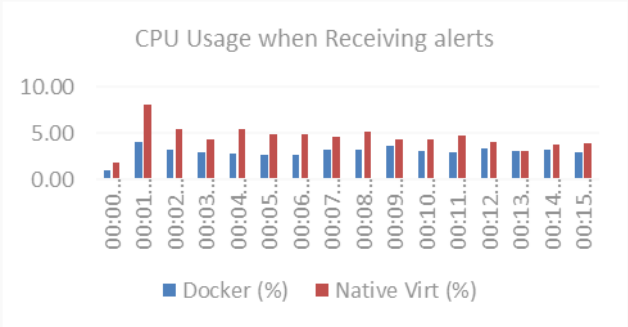

Figure 12. Comparison of CPU Usage of both systems when receiving an alert

Figure 12 shows a comparison of the CPU usage test results for both systems in the condition of receiving alerts within 15 minutes. The graph presented shows that CPU usage on the system using containers has an average usage of 2.98% which is lower than conventional virtualization at an average of 4.55%. This comparison does not have a significant effect on the performance of the two systems due to relatively low utilization, below 10% CPU usage. However, from this it can be seen that the Docker Container implementation provides CPU resource efficiency by showing a lower average usage.

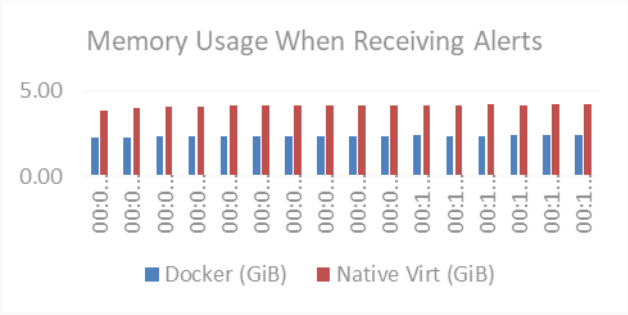

Figure 13. Comparison of Memory Usage of the two systems when receiving an alert

Figure 13 shows a comparison of the memory usage test results of the two systems in the condition of receiving an alert within 15 minutes. From the graph presented, it can be seen that there is quite a significant difference in the memory usage of the two systems, showing that the memory usage of the system that uses containers is average. usage at 2.34 GiB which is lower than conventional virtualization at an average of 4.13 GiB. Here it can be seen that the Docker Container implementation provides memory resource efficiency with lower average usage.

To summarize the performance comparison under the simulated attack load, the key metrics are presented in the table below. The values represent the average and peak utilization observed during the 15-minute testing period.

TABEL 2.
PERFORMANCE COMPARISON

Metric	System	Average (Under Load)	Peak (Maximum)
CPU Usage	Docker Container	2.98%	6.46%
	Conventional Virtualization	4.55%	10.93%
Memory Usage	Docker Container	2.34 GiB	2.39 GiB
	Conventional Virtualization	4.13 GiB	4.18 GiB

Tabel 2 shows a significant difference in resource usage, where the system using Docker containers is consistently more efficient in both CPU and memory utilization.

IV. CONCLUSION

Design of container implementation in security information management system infrastructure using Docker as a container platform and Wazuh as a security information management system platform. In this implementation, each Wazuh service is defined and configured using Docker Compose. The Docker Compose configuration that has been designed runs three containers, each of which runs the wazuh service, namely wazuh-indexer, wazuh-manager, and wazuh-dashboard.

Based on performance testing carried out on both systems, namely the system that uses Docker Containers and Conventional Virtualization, the system that uses Docker Containers shows lower Memory and CPU usage compared to Conventional Virtualization, this makes the container implementation carried out able to provide advantages in memory resource efficiency and CPU.

REFERENCES

[1] Fitri Nova, M. D. Pratama, and D. Prayama, "Wazuh sebagai Log Event Management dan Deteksi Celah Keamanan pada Server dari Serangan Dos," JITSI: Jurnal Ilmiah Teknologi Sistem Informasi, vol. 3, no. 1, 2022, doi: 10.30630/jitsi.3.1.59.

[2] [Docker Inc., "Docker Documentation | Docker Documentation," Docker Documentation, 2020, Accessed: Jan. 29, 2024. [Online]. Available: https://docs.docker.com/.

[3] Dua, Rajdeep, A. Reddy Raja, and Dharmesh Kakadia. "Virtualization vs containerization to support paas." 2014 IEEE International Conference on Cloud Engineering. IEEE, 2014. doi: 10.1109/IC2E.2014.41.

[4] R. Felani, M. N. Al Azam, D. P. Adi, A. Widodo, and A. B. Gumelar, "Optimizing Virtual Resources Management Using Docker on Cloud Applications," IJCCS (Indonesian Journal of Computing and Cybernetics Systems), vol. 14, no. 3, 2020, doi: 10.22146/ijccs.57565.

[5] F. Mulyadi, L. A. Annam, R. Promya, and C. Charnsripinyo, "Implementing Dockerized Elastic Stack for Security Information and Event Management," in InCIT 2020 - 5th International Conference on Information Technology, 2020. doi: 10.1109/InCIT50588.2020.9310950.

[6] Rad, Babak Bashari, Harrison John Bhatti, and Mohammad Ahmadi. "An introduction to docker and analysis of its performance." *International Journal of Computer Science and Network Security (IJCSNS)* 17, no. 3 (2017): 228.

[7] *Sollfrank, Michael, Frieder Loch, Steef Denteneer, and Birgit Vogel-Heuser. "Evaluating docker for lightweight virtualization of distributed and time-sensitive applications in industrial automation." IEEE Transactions on Industrial Informatics 17, no. 5 (2020): 3566-3576.*

[8] S. Sebastio, R. Ghosh, and T. Mukherjee, "An Availability Analysis Approach for Deployment Configurations of Containers," IEEE Trans Serv Comput, vol. 14, no. 1, pp. 16–29, 2021, doi: 10.1109/TSC.2017.2788442.

[9] G. N. Schenker, The ultimate Docker container book build, test, ship, and run containers with Docker and Kubernetes. Packt Publishing Ltd., 2023.

[10] E. N. Preeth, J. P. Mulerickal, B. Paul, and Y. Sastri, "Evaluation of Docker containers based on hardware utilization," in 2015 International Conference on Control, Communication and Computing India, ICCC 2015, 2016. doi: 10.1109/ICCC.2015.7432984.

[11] G. N. Schenker, The ultimate Docker container book build, test, ship, and run containers with Docker and Kubernetes. Packt Publishing Ltd., 2023.

[12] Wazuh Inc., "Wazuh Documentation." Accessed: Jan. 23, 2024. [Online]. Available: https://documentation.wazuh.com/current/index.html.

[13] A. Makris et al., "Streamlining XR Application Deployment with a Localized Docker Registry at the Edge," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2023. doi: 10.1007/978-3-031-46235-1_12.

[14] S. R. Dira, M. Arif, and F. Ridha, "Monitoring Kubernetes Cluster Menggunakan Prometheus dan Grafana," Proceeding Applied Business and Engineering Conference, no. November, 2022..

[15] F. I. F. Farrel, Is Mardianto and Adrian Sjamsul Qamar, "Implementation of Security Information & Event Management (SIEM) Wazuh with Active Response and Telegram Notification for Mitigating Brute Force Attacks on The GT-I2TI USAKTI Information System," Intelmatics, vol. 4, no. 1, 2024, doi: 10.25105/itm.v4i1.18529.

[16] P. S. S. Patchamatla, "Security Implications of Docker vs. Virtual Machines," *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 12, no. 09, Jun. 2023, doi: 10.15680/IJIRSET.2023.1209003

[17] Ardi, Noper, S. Supardianto, and Ahmadi Irmansyah Lubis. "Predicting missing value data on IEC TC10 datasets for dissolved gas analysis using tertius algorithm." *Journal of Applied Informatics and Computing* 7, no. 1 (2023): 50-56.

[18] Yang, Hubin, et.al, "REDB: Real-time enhancement of Docker containers via memory bank partitioning in multicore systems. *Journal of Systems and Software*, vol. 151, p. 103135, 2024

[19] Soldani, J., & Brogi, A., "Docker-based models for engineering IoT systems," *Journal of Systems and Software*, vol. 188, p. 111270, 2022

[20] Oliveira, Asiss T et.al, "Analysis of SR-IOV in Docker containers using RTT measurements" *Journal of Systems and Software*, vol. 228, p. 107961, 2024

[21] Gangula, Rekha, et.al, Integration of dynamic Docker containers and kuberneters with advanced cloud and Internet of Things", Materialstoday:Proceeedings, Vol. 80 2023, Pages 3476-3480