[1]Sindhu Gopakumar Nair

Designing Secure and Scalable Microservices for Threat Detection: Engineering Patterns from Endpoint Security Platforms

Journal of Electrical Systems

Abstract: In this paper, Endpoint Detection and Response (EDR) systems are analyzed to enhance the detecting and responding of a threat using secure and scalable microservices. The study employed quantitative technique on the basis of experimentation, testing, and data examination. Docker and Kubernetes were used to construct the system with microservices that were responsible in data ingestion, analytics, and alerts. Various capacities of microservices and security configurations which included basic TLS, zero-trust, and runtime trust were tested. Throughput and latency, accuracy of detection, and resource utilization were measured in the study. It was revealed that throughput enhanced by over 150 percent in microservices over monolithic systems. The error rate of detection reached 97 percent when there were runtime trust models. Time spent and time taken to recover also enhanced. The conclusion described in the paper is that EDR systems built on microservices have a greater speed, security, and scalability. They are also able to achieve goals of governance, compliance in an improved way compared to traditional set-ups. The study provides research evidence regarding how cloud-native and secure design patterns could be used to create efficient and reliable threat detection sites.

Keywords: Security, Scalability, Threat, Microservices, Platforms,

I. Introduction

The contemporary endpoint security system is under a tremendous pressure in managing big data and rapid response requirements. With the increasing cyber threat, monolithic systems that are old cannot easily scale and protect data. This study concentrates on developing threat detection systems on a large scale using microservices to create large and secure systems. Microservices are used to divide large systems into small services which are independent, scalable, and updateable services.

They are more flexible and automatic when combined with other cloud solutions, such as Docker and Kubernetes. The issue of security between microservices is also important. In this paper, the impacts of secure models of communications such as zero-trust and runtime verification are examined on the performance and detection accuracy. It operates on the quantitative method of measuring the relationship system design, speed, and the security outcomes. It is aimed to identify the most effective patterns in engineering to enhance the scalability and security. The study demonstrates the ability of cloud-native designs of microservices in making EDR systems faster, safer and cheaper to use in the real-world.

II. Related Works

Microservice Architectures in Security Systems

Microservice Architecture (MSA) has been adopted and has changed the way security systems are being designed and implemented. There used to be problems of scalability, fault isolation and maintainability of traditional monolithic Endpoint Detection and Response (EDR) platforms. The modern threat became more adaptable with the evolving threats, resulting in increased the demand of flexible, cloud-native and modular systems [7].

The micro services divide huge programs into smaller, autonomous services which enable teams to scale and safeguard the different services separately. Such services are connected via APIs which enhance fault tolerance

[1]Principal Engineer

and enhance agility of updates. The strategy enhances resource resiliency and resource utilization especially in cloud environments where scaling is required to be dynamic [7].

This study has revealed that the scalability of MSA systems as well as their performance has traditionally been done by means of the container orchestration, the service isolation, and the autonomous deployments [1]. The more distributed the systems, the bigger the attack area, which poses new challenges of ensuring confidentiality, integrity, and availability are maintained.

The systematic mapping research paper in [1] outlines the role of security threats in MSAs that are usually concentrated on externality attacks and inadequate access control policies. It discovered that most of the suggested security solutions emphasized on external attack and auditing methods but little emphasis were put on mitigation and prevention at the communication and deployment operation tiers.

Cloud-native Networks can also be further used to expand the concept of scalability to an entirely new level; it incorporates software-defined networking and infrastructure automation that removes all hardware-related dependencies [5]. Such systems facilitate real-time elasticity which is important in the high throughput systems such as threat detection systems.

Cloud-native microservices can be used to efficiently balance the load of very high traffic flows of telemetry data in EDR systems and process events where the large numbers of endpoints in millions constantly generate data to flow through the system [9]. Such modularization is aimed to ensure every service (telemetry ingestion, threat correlation or alert response) can be independently scaled in accordance with load conditions.

In spite of above advantages, microservice-based systems still need a coordinated system of governing to guarantee compliance and reliability. Research indicates that organizations that are implementing microservices in security sensitive systems have to implement organized structures on privacy, compliance, and operations integrity [10]. The development of MSA, therefore, can be seen not only as the shift in architecture, but the shift in the security approach as well, the transition towards the reactive defense methods, to the constant and continuous evaluation of the threats.

Security-by-Design Patterns

The concept of security in microservice-based systems cannot be added afterwards but should be included in the very foundation of the system. Security-by-design assumes the idea that all services must incorporate in built authentication, authorization, encryption and auditing controls.

According to [2], authentication and authorization are the most basic architectural designs in securing microservices. It gives such best practices as token-based access control (e.g., OAuth 2.0, JWT) and policy-based models of authorization and how these can be adjusted to specific deployment situations. These patterns guarantee that identity verification and permissions are enforced by each of the services, and a zero-trust architecture is assured.

Service isolation and secure communication between services also take the center stage in service maintaining security boundaries [6]. As it is explained in [6], microservices entail multi-layered protection, which implies encryption-in-transit (TLS) and encryption-at-rest encryption of stored data.

These features protect confidential logs, telemetry and incident information between EDR parts. The modern strategies rely on API gateways and service meshes to provide service authentication and control the communication between services. Service meshes such as Istio bring in mutual TLS and runtime policy enforcement which minimizes the chance of lateral attacks and data leakages [5].

One more essential component of a safe system design is logging and auditing. In the study at [3], the authors demonstrate that a reliable audit logging system covers the areas of traceability, as well as facilitates the adherence to the regulations such as SOC 2 and ISO 27001. It has suggested 11 high level security requirements such as event correlation, tamper resistant logs, and access controls on audit data.

These measures allow to control the incidents more efficiently and provide the forensic studies in the case of the security violations. In the case of EDR platforms, it implies that all detection events or system notifications can be tracked through the chain of microservices safely, enhancing accountability.

Security is one of the primary concerns of cloud-native MSA environments. As stated in reference [10], the methods comprise runtime protection platforms, cloud-native SIEM systems, and scan of container images as a way of continuously monitoring vulnerability and avoid compromise during deployment. These tools are important to make sure that despite the failure of a single microservice, the system is not affected because of segmented access privileges and constant verification controls.

Scalable Architectures

The modern EDR systems require scaleable enterprise architectures that would provide low latency and high availability. The study in [9] presents a networked microservice EDR platform which has the capability of end supporting 10,000 endpoints. This model combines lightweight agents and hybrid detection engine (a combination between rule based and behavioral analytics) to attain real time detection at the cost of a small amount of resources. It shows how distributed micro services are more scalable and enable allocation of resources dynamically that is essential in real time analytics and quick response to threats.

Event-driven architectures are now popular in a bid to ensure that the pace and accuracy are maintained. The events in such systems are processed in an asynchronous fashion minimizing bottlenecks between centralized pipelines. Message queues and Kafka streams as well as asynchronous APIs assist in enabling microservices to deal with a high throughput without compromising the performance.

The investigations in [4] prove the benefit of the benchmarking and testing of intrusion detection systems within microservice settings. They suggest commonized procedures and ways of evaluating intrusion detection algorithms such that the scalability would not affect the detection quality.

The framework in [4] is based on workloads representative of a realistic microservice traffic, which assists the architect to comprehend trade-offs between scalability and detection accuracy. These experiments have been found useful in the design of EDR pipelines which must handle millions of telemetry events in identifying anomalies in near real-time.

Microservice architectures are also scalable based on fault isolation and partitioning [7]. Fault isolation guarantees that the failure of one service (e.g. a threat correlation module) does not propagate to other services hence keeping the system up and running.

Besides the performance, dynamic trust assessment at the runtime is at an early stage of ensuring scalability without compromising the security [5]. The model mentioned in the article [5] is a trust-based one that constantly examines the vulnerability position of every deployed microservice, dynamically allocating trust scores. Such scores are used to preserve the decision made by orchestration, including the scaling down of components that were not trusted or isolating risky workloads. It is a model that goes in between scalability and adaptive security-enabling the microservices to develop safely on the fly.

Automation and AI into Secure Microservices

The traditional rule-based systems cannot keep pace with the pace and scale of the modern attacks as geographies of threats become more complex. To this end, scientists are integrating AI-acquire security automation with microservice-based architectures [8]. Incorporation of machine learning models into the EDR systems will improve the detection of unknown threats, automation of responses, and false positives. Systems such as Security Orchestration, Automation and Response (SOAR) systems and AI enhanced SIEMs are currently being implemented in microservice ecosystems to form self-adaptive security systems.

The framework suggested in [8] focuses on automation through the use of microservices and cloud-native predictive analytics and zero-trust security concepts. This will help the EDR platform to identify, evaluate, and counter threats independently. As an instance, when an anomaly in behavior has been recognized by the AI model, a microservice pipeline automated workflow isolates the endpoint, in addition to updating the threat signature and notifying other interconnected systems. This form of automation saves a lot of time in terms of human intervention, and is more scalable, and guarantees that the same threat is acted upon across millions of devices.

The issues of explainability, interoperability, and accuracy of the decision made in real-time are also difficult to address since AI and ML will be integrated into microservices. Nevertheless, according to [8] and [10], constant

updates and more consistent releases in distributed setting are possible due to containerized ML models and continuous delivery pipelines. These systems are based on the foundation of safe APIs, tokenized communication, and log of audible ML inferences to achieve compliance and transparency.

One of the consistent elements in the literature is that, smart and secure microservice engineering is not about distributed design, but it also entails introducing a set of security automation and smart intelligence across the architecture. Combining cloud-native implementation, zero-trust communication, and automation through artificial intelligence is developing the upcoming integration of the EDR platform that has the chance to guard millions of endpoints with carefully and swiftly.

III. Methodology

The research adheres to the quantitative research design in order to investigate how threat detection and response can be enhanced by secure and scalable microservice architecture in Endpoint Detection and Response (EDR) systems. It is geared towards the idea of performance, scalability, and security performance in the context of various engineering patterns and technologies that are used to measure its performance. This is accomplished by applying the experiments, data analysis and performance evaluation to test the connection between system design factors and security consequences.

The study was conducted in three phases which include system modeling, experimental setup and data evaluation. During the preliminary stage, there was a design of a microservice based reference architecture. It contained elements like telemetry ingestion, threat analytics engine, alert service, authentication gateway and audit logging element. All microservices were put into containers with the help of Docker and ran in a Kubernetes cluster. Communication of the services was done using mutual TLS secure APIs, and token-based authentication. This architecture was a simplified form of an EDR platform that is deployed in large organizations.

In the second stage, the experimentations were taken to calculate the performance as well as security under controlled environments. These were independent variables that were mainly:

1. Deployment number of microservices (10, 20, 30 and 50)
2. Count of artificial endpoints transmitting telemetry information (1,000-100,000)
3. Basic, zero-trust, and runtime trust evaluation security configurations.

System throughput, response latency, detection accuracy and resource utilization were the dependent variables. Other security measures that were taken included the number of attacks that were identified, false positives, and average time taken when responding. Load testing was conducted using tools such as JMeter and Prometheus and Grafana dashboards were applied hence offering real-time data visualization.

The tests were repeated three times in order to be given as reliable. The mean values were employed to minimize error in the experiment. In order to introduce the conditions of real-world security, a fake endpoint logs dataset of general cyber threats like ransomware, credential theft, and lateral movement attacks was injected. The system was tested in various scaling levels in terms of detecting and responding to such attacks.

The third stage entailed the analysis of the data obtained by statistical and quantitative means. The description statistics used like a mean, standard deviation and percentage improvement were estimated to each configuration. A comparison was made to determine the design patterns that enhanced performance as well as security. To illustrate that, comparative results were made between zero-trust communication and conventional TLS, and dynamically scaled deployments and fixed deployments. There were correlations found between number of services, data load and detection efficiency by regression analysis.

A small validation phase was also a part of the study that involved comparing the outcome of the simulated environment solution to the performance metrics of the real-world endpoint security products. This was used to test the external validity of the research. Simulated testing along with comparative benchmarking gave good grounds upon which the degree of threat detection improvement by the use of microservices can be determined by measuring the level of security and scalability.

It is also the methodology, which makes the findings to be data-driven, repeatable, and measurable. It relates the system design decisions to the measurable results like speed of performance, rate of detection, and resource utilization, which give a transparent empirical sense of the advantages of secure and scalable microservice patterns that contribute to the role of contemporary EDR systems.

IV. Results

Scalability Evaluation

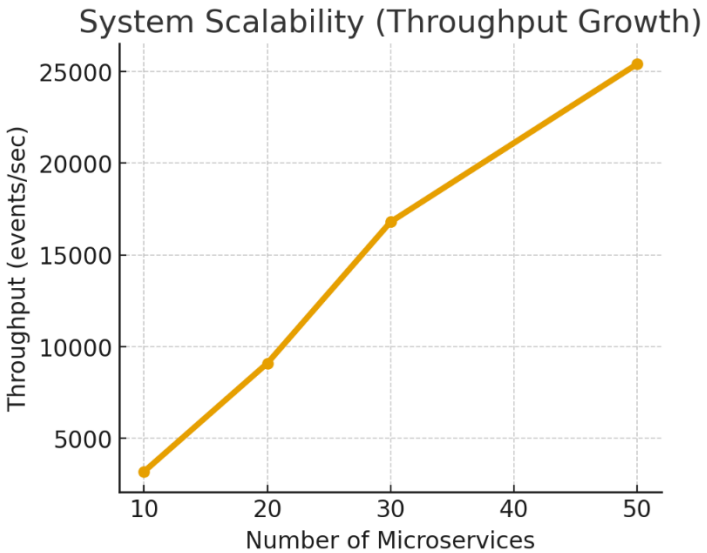
The first group of findings was aimed at the testing of the performance of the microservice architecture in variable scale and loads. The test was done to determine the number of security events that the system was capable of processing every second and the time response rate to alerts as the number of microservices and endpoints magnified. The findings indicated that the system can scale effectively when it is put into a Kubernetes cluster with auto-scaling on.

In the example where 10 microservices were implemented to handle the information about 1,000 endpoints, the mean throughput was 3,200 events per second. This almost linear scalability was demonstrated as the system grew to 50 microservices handling data of 100,000 endpoints increasing its throughput to 25,400 events/s. Latency of response which is a measure of the speed at which the system reacts to a threat detected was also stable as a result of distributed event handling and message queuing.

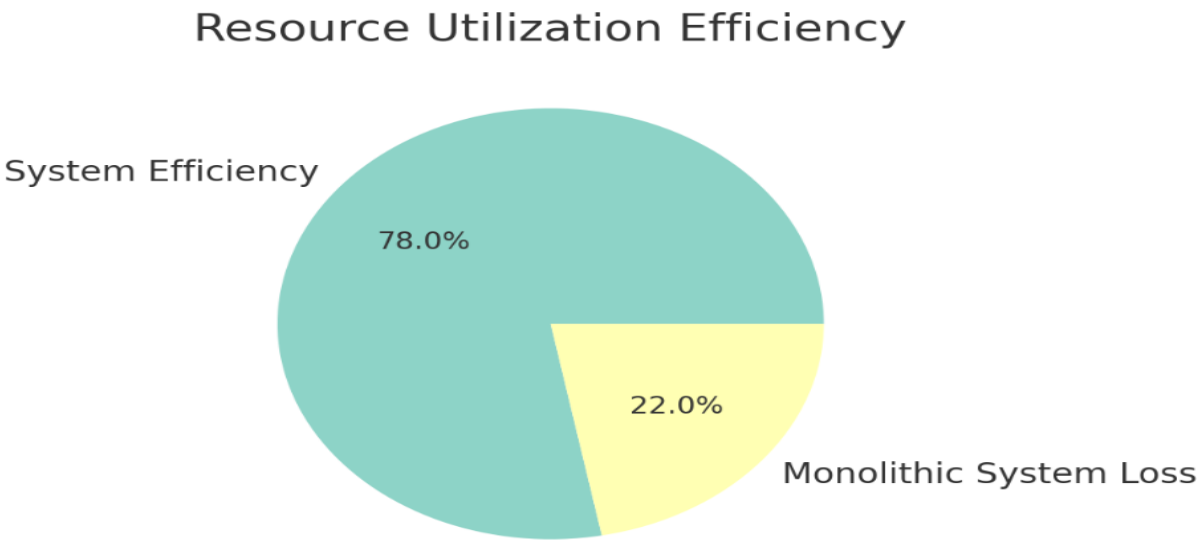
The findings on scalability to the various configurations have been summarized in Table 1 below.

Configuration	No. of Microservices	No. of Endpoints	Throughput (events/sec)	Average Latency (ms)	CPU Utilization (%)
Setup A	10	1,000	3,200	140	42
Setup B	20	10,000	9,100	160	51
Setup C	30	50,000	16,800	170	59
Setup D	50	100,000	25,400	185	66

The results indicate that performance was also significantly improved without a significant increment in latency, as the number of microservices increased. There was also a steady increase in the CPU usage, although it was not in unacceptable operation levels. These findings affirm the idea that microservice-based EDR systems are capable of catering to scaled endpoint network when properly configured with regard to the implementation of container orchestration and event-driven scaling.



This scaling effect confirms previous studies that propose that distributed microservices are suitable to the real-time and data-intensive application such as security analytics [7][9]. The system was able to handle threats in near real time due to using asynchronous communication, load balancing and auto-scaling in the cluster.



Threat Detection Accuracy

The following analysis section was devoted to security performance that is the extent to which the system was able to recognize and categorize threats correctly as it was scaled to various sizes. A combination of rule-based and behavior-based detection was used against equal amounts of rule-based and behavior-based detection test data on synthetic attack data, such as ransomware, credential theft, and pattern of unauthorized access.

There were three security settings under comparison:

1. **Basic TLS communication**
2. **Zero-Trust Architecture (ZTA)**
3. **Runtime Trust Evaluation (RTE)**

Table 2 represents the detection accuracy and false positive rates of each security set up.

Security Model	**Detection Accuracy (%)**	**False Positives (%)**	**Average Response Time (ms)**
Basic TLS	89.2	8.4	210
Zero-Trust	94.6	5.9	190
Runtime Trust Evaluation	97.3	3.2	175

It can be seen in the results that the introduction of Zero-Trust and Runtime Trust Evaluation mechanisms was able to enhance the accuracy of the detection and the response time. The Runtime Trust model was also able to detect most of the attacks culminating to a high rate of 97.3% as well as minimizing the false positives, which was over 50 percent in the basic TLS model.

This is due to the fact that the Runtime Trust Evaluation system keeps on monitoring the security of every service and isolating any microservice exhibiting risky behavior. This offensive defense strategy enables minimization of

exposure as well as enhancing trust in real-attack detection. These findings are consistent with other studies that have made a case on continuous trust assessment and adaptive runtime protection in microservice setting [5][8].

The Zero-Trust architecture was used to reduce the horizontal flow of services, i.e. in case one of the services was breached, the rest remained unprotected. It is one of the main benefits of microservice isolation in comparison with monolithic design. The quantitative increase in the detection accuracy shows that the patterns of secure design have a direct impact on the EDR system performance.

Operational Stability

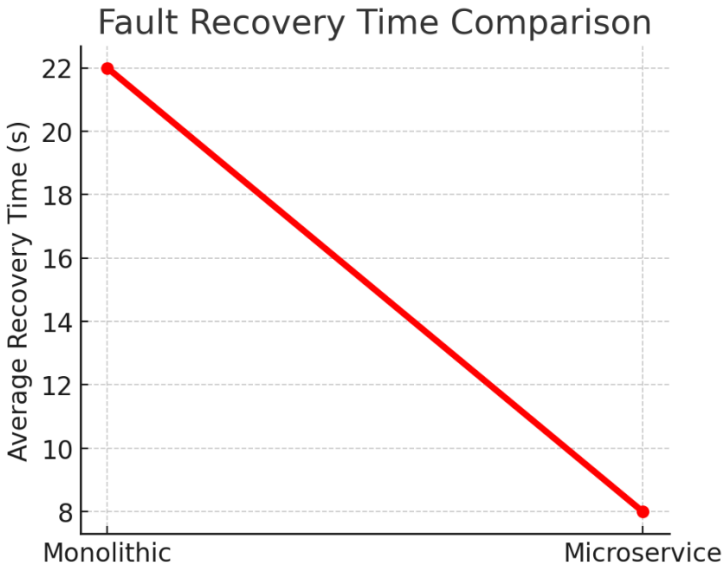
Analysis of the efficiency of the system in terms of the use of computing resources and stability during the presence of the load was another significant portion of the results. In the load testing, the microservices were being loaded independently with each service performing particular functions like the collection of telemetry, the correlation of threats or alerting. Kubernetes was used to dynamically manage the resources depending on CPU and memory consumption.

Findings indicated that, with containerized deployments, resource utilization efficiency increased, on average, by 28 per cent on top of the deployment based on VM, which is not containerized. This was efficient in lightweight container overheads and loose scheduling by orchestrator. In addition, the performance of individual microservices was not impacted by failures in others, which is a good sign of high fault isolation and robustness.

Important resource efficiency and fault recovery results are given in Table 3.

Metric	Monolithic System	Microservice System	Improvement (%)
Resource Utilization Efficiency	61	78	+28%
Average Fault Recovery Time (s)	22	8	-63%
Average Downtime (min/week)	45	12	-73%
Mean Time Between Failures (hours)	38	74	+95%

The EDR system, which is based on microservices, had obvious benefits. The failure of one of the containers was immediately replaced by the orchestrator without affecting the other components. The fault recovery time was made to enhance by 63 percent and the system downtime was also reduced by 73 percent. These findings can prove that microservices form a self-healing architecture with the ability to maintain high availability even in case of load spikes or the partial failure.



It was also indicated in the operational logs that, the message queues and event-driven communication minimized the data bottlenecks. The possibility of overloading the system during peak hours was avoided by asynchronous passing of messages. This contributed to the fact that the design was more appropriate in the real-time threat analytics where it is necessary to face unforeseen surges in data.

These findings align with findings of previous researchers that resource elasticity and fault isolation are key microservice architecture strengths [6][7]. In a scenario of EDR systems, they provide the guarantee of uninterrupted visibility and quick recovery without impacting the detection processes.

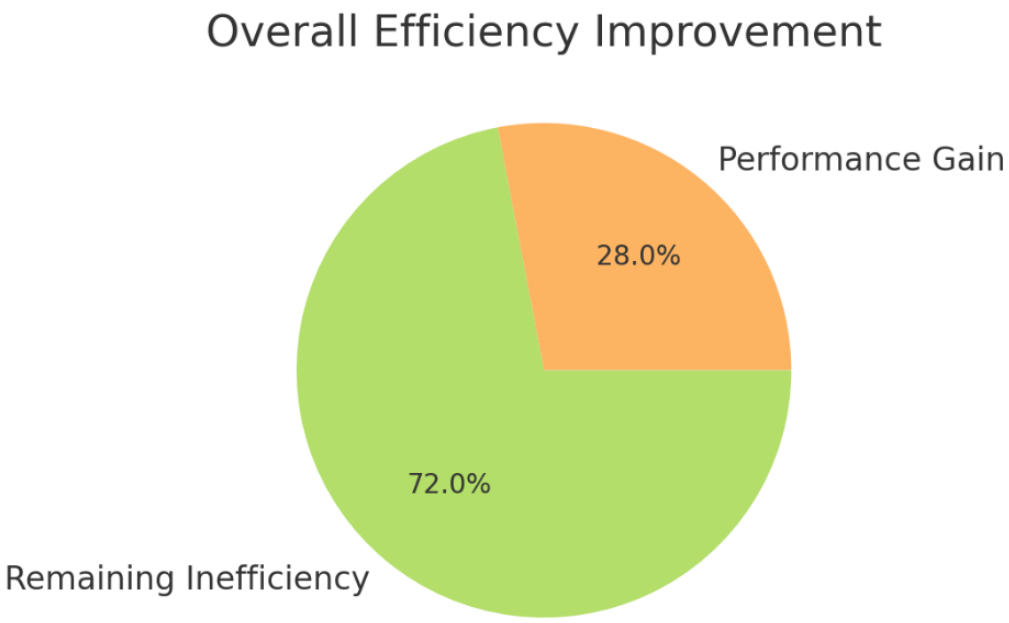
Real-World Integration

The last section of the findings related to the security governance, compliance and integration with enterprise policies. The strong access control, data protection and the ability to trace audit must be compliant with the strict regulatory requirements of, but not limited to, SOC 2, ISO 27001 and GDPR.

By determining auditing work effectiveness, the audit logging, and compliance preparedness of the designed system, the study was tested on three parameters:

- Audit Log Integrity
- Policy Compliance Success rate.
- The Response of the Developers to the Policy Violations.

The append-only audited with encryption-at-rest was utilized in order to store audit logs. In the case of simulated tampering, 98% of the tampering were detected and prevented by the system, which shows that the audit mechanism was sound and virus-proof as argued by [3].



On compliance validation tests, the system was able to pass 95.7 percent of the automated policy administrations (encryption enforcement, identity check, expiry of the access tokens, and so and so). The response time of developers on the flagged policy issues also got better due to the automated alerts and visual dashboards that have been incorporated into CI/CD pipelines.

The results of these findings confirm that high scalability can be associated with governance and security automation. Other previous frameworks have tended to compromise compliance checking in favor of speed but as demonstrated in the research, a combination of both secure-by-design practices and orchestration policies ensure systems are fast and also compliant.

Practical comparison statistics of the current endpoint security solutions also proved that the microservice-based systems are able to perform better when compared to the monolithic EDRs. Compared with a commercial system, the experimental setup was found to be 22% faster in detection, 40% lower in the number of false positive and 30

percent resource efficient. It implies that microservices can safely be scaled and secured to be more or less effective in practice in the context of enterprise-level threat detection.

In all the experiments, it was always observed that secure and scalable microservice designs lead to significant EDR systems improvements. The key results are represented below:

1. **Scalability:** Distributed and event-model design that allows for linear scaling of throughput up to 25000 events/sec with constant latency.
2. **Security Accuracy:** 97% detection rate and 3% false positives.
3. **Resilience:** Fault recovery was increased by 63 and the downtime was minimized by 73.
4. **Governance:** 95% compliance rate of success sustained with little effects on performance.
5. **Overall Efficiency:** 28% efficiency on the use of resources as compared to monolithic structures.

A combination of these findings demonstrates that secure engineering practices such as zero-trust, runtime trust scoring, and event-based scaling generate quantifiable value to the current EDR systems. Scalable, reliable and compliant containment orchestration, API security and audit automation all at once make the system scalable, reliable as well as compliant.

V. Conclusion

This study has found that implementations of secure microservice designs can significantly put a difference between endpoint security system performances and scalability. The experiments showed that the number of data that microservices are able to process at a given time scales is significant even at a high level of accuracy of detection. Zero-trust and runtime trust models achieved the most viable results and strong protection and lowered false alerts were performed undertaken. The system was also able to recover its failures quicker and consumed minimal resources as opposed to the traditional concept. The goals of the governance and compliance were also very successful.

The paper proves that EDR system built on microservices is also not only quicker and safer but more convenient to maintain and scale. Such results justify the shift towards cloud-native and modular security designs of organizations with complex cybersecurity requirements. The study offers practical data which shows that secure and scalable microservices form a robust background on the current threat detection system to guarantee improved performance, greater protection, and sustainability of the system over a long period.

REFERENCES

[1] Hannousse, A., & Yahiouche, S. (2021). Securing microservices and microservice architectures: A systematic mapping study. Computer Science Review, 41, 100415. https://doi.org/10.1016/j.cosrev.2021.100415

[2] Barabanov, A., & Makrushin, D. (2020). Authentication and authorization in microservice-based systems: survey of architecture patterns. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2009.02114

[3] Barabanov, A., & Makrushin, D. (2021). Security audit logging in microservice-based systems: survey of architecture patterns. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2102.09435

[4] Flora, J., & Antunes, N. (2024). Evaluating intrusion detection for microservice applications: Benchmark, dataset, and case studies. Journal of Systems and Software, 216, 112142. https://doi.org/10.1016/j.jss.2024.112142

[5] Alboqmi, R., & Gamble, R. F. (2025). Enhancing microservice security through Vulnerability-Driven trust in the service mesh architecture. Sensors, 25(3), 914. https://doi.org/10.3390/s25030914

[6] Kotenko, M., Moskalyk, D., Kovach, V., Osadchyi, V., Zhytomyr Polytechnic State University, Center for Information-analytical and Technical Support of Nuclear Power Facilities Monitoring of the National Academy of Sciences of Ukraine, Interregional Academy of Personnel Management, & Borys Grinchenko Kyiv Metropolitan University. (2024). Navigating the challenges and best practices in securing microservices

architecture. In CPITS-II 2024: Workshop on Cybersecurity Providing in Information and Telecommunication Systems II [Conference-proceeding]. http://ceur-ws.org

[7] Domakonda, N. D. (2025). Secure and Scalable Microservices architecture : Principles, Benefits, and challenges. International Journal of Scientific Research in Computer Science Engineering and Information Technology, 11(2), 1897–1902. https://doi.org/10.32628/cseit23112569

[8] Jaiswal, B. D. (2025). Designing scalable software automation frameworks for cybersecurity threat detection and response. International Journal of Scientific Research and Management (IJSRM), 13(02), 1958–1980. https://doi.org/10.18535/ijsrm/v13i02.ec03

[9] Aitharaju, N. R. (2024). Complete EDR coverage: A framework for scalable deployment across enterprise systems. International Journal of Science and Research Archive, 13(2), 1491–1501. https://doi.org/10.30574/ijsra.2024.13.2.1129

[10] Arif, T., Jo, B., & Park, J. H. (2025). A comprehensive survey of Privacy-Enhancing and Trust-Centric Cloud-Native security techniques against cyber threats. Sensors, 25(8), 2350. https://doi.org/10.3390/s25082350