International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V72I10P114

Volume 72 Issue 10, 86-93, October 2024

Original Article

Overcoming Security Obstacles in Serverless Function-as-a-Service (FaaS) for Healthcare Insurance

Sanjeev Kumar

Independent researcher, SME in Cloud Engineering, Georgia, USA.

Corresponding Author : sanjeevkumar.sk@ieee.org

Received: 30 August 2024 Revised: 03 October 2024 Accepted: 18 October 2024 Published: 30 October 2024

Abstract *- In recent years, serverless computing, particularly FaaS, has gained much popularity as a method by which developers can develop and publish their code without having to manage any underlying infrastructure. With these conveniences and scalability opportunities come a particular set of security challenges: function-level vulnerabilities, insecure APIs, data leakage risks, improper resource permissions, and bad monitoring practices. Furthermore, the stateless nature of FaaS combined with shared environments in the cloud increases the number of attack vectors, which include injection attacks, DoS, and privilege escalation. This paper searches for general security challenges of serverless applications, especially FaaS, and provides a detailed review of best practices available to mitigate the risks. The studies are analyzed based on case study data, and the findings from security testing tools, such as OWASP ZAP and Burp Suite, which have identified the vulnerabilities of the application and measured the effectiveness of various security practices, are considered. These tools are applied in a simulated FaaS environment, and the findings are drawn from the attack frequency impact of security measures on system performance, so demonstrating how best practices such as least privilege access, API security, and encryption can really make a difference in security outcomes. Risks will be reduced, and compliance with modern security standards will be upheld by adopting a holistic, security-first approach to the design of serverless applications. This paper provides an overall roadmap for building secure and efficient FaaS with real-world examples and empirical evidence.*

Keywords *- Serverless Security, Function-as-a-Service (FaaS), Cloud Security, API Security, Secure architecture.*

1. Introduction

The cloud and, indeed, serverless architecture has significantly changed the landscape of software development and deployment. Recent trends reflect the fast growth of serverless architectures. The Function-as-a-Service (FaaS) model is one of the bold features of serverless computing, which allows developers to focus only on coding and not care about the underlying infrastructure. This shift has been instrumental in creating highly scalable, cost-effective, and agile applications [1]. FaaS allows applications to scale automatically in response to demand while charging only for the actual usage of compute time, making it extremely appealing to businesses seeking to minimize overhead costs and reduce operational complexity. However, though it has massive advantages, FaaS brings into the equation equally massive security risks. In a serverless environment, the attack surface is inherently more different than in traditional architecture [2]. It means that with a lack of control by developers over the infrastructure, application functions end up being insecure and data integrity compromised. Some of these include insecure APIs, function-level vulnerable points, data leaks and improper permissions, among others. Serverless computing comprises an attack surface that is segmented into smaller, more modular components than in other traditional systems, thus requiring different security approaches. Since serverless platforms are, by nature, multi-tenanted shared resources in a cloud environment, this exposes applications to a myriad of threats [3]. For one, it makes privilege escalation and DoS attacks especially easy when implementing a serverless model. Statelessness is another product of serverless functions, which causes problems related to keeping session security and isolating runtime for functions. Most serverless applications have third-party libraries and APIs that open up the threat of vulnerabilities in third-party code [4].

This paper addresses the critical issues using an all-inclusive analysis of security threats in FaaS-based serverless applications. The paper will outline major security threats, vulnerabilities, and attack vectors and propound best practices to mitigate them. In doing so, it discusses how developers can adopt security-first design principles to make their applications resilient to modern security threats. In addition, an architectural model for safe serverless deployment will be proposed, and related security analyses will outline how specific protection measurements may reduce the vulnerabilities in FaaS. Laying a foundation based on both theoretical and practical perspectives of serverless security, this paper aims to contribute knowledge to the sea of growing

research surrounding the development of serverless applications. This paper will elaborate upon additional security practices brought about by the new information security practices: encryption, least-privilege access, and runtime monitoring under real-world scenarios. It should also put emphasis on maintaining a security posture on proactive terms by continuous scanning and monitoring for vulnerabilities. Serverless computing will only continue to proliferate, and it needs to address these new challenges head-on. Security practices should be employed in the development stage of organizations' serverless applications but should also spread throughout their lifecycles. This paper will guide developers, security professionals, and businesses on how to adopt serverless technology easily while at no point compromising on security. This research will prove handy in the mitigation of risks associated with serverless applications, providing all its findings with complete analysis and practical recommendations on the same and, therefore, help businesses understand how to properly benefit from FaaS while keeping up a good level of security posture [5].

2. Review of Literature

In the past two years, the literature on serverless computing has grown, and many studies have underlined the benefits, such as reduced costs, scalability, and ease of deployment. Meanwhile, much of the literature has underlined security vulnerabilities that are specific to these serverless models, particularly within the Function-as-a-Service environment [6]. Thus, the onus now appears to go down the layers to the application, whose security threats the developers will have to address directly in the function code at that layer. Some critical studies enumerate for themselves some of the issues when it comes to securing APIs- the main entry or access points into the system for communication between serverless functions [7]. APIs are critical entry points because there is a likelihood of injection attacks and privilege escalation occurring at that level. In addition, being stateless, FaaS functions further complicate matters like session persistence or protection of data, for example, because it will be very hard to apply the usual secure aspects like server-side authentication or authorization in a stateless way [8]. This is another risk of serverless functions, by their own literature outlining the risks of third-party libraries used within serverless functions. Serverless applications are fundamentally very modular; developers rely very much on third-party libraries just to speed up development processes. However, such use leaves the software at risk of misuse by malicious actors because the libraries are not trusted. As a result, supply chain security has become a recent point of focus in research into serverless security. Another area that existing literature addresses involves the shared responsibility model between cloud service providers and users. Here, while the provider guarantees securing the infrastructure, the user is responsible for securing their applications. This model has led to several security best practices, such as the principle of least privilege; it also ensures minimal attack surface on serverless functions [9]. Developers are encouraged to make minimum permissions for functions to be able to perform their tasks without the risk of privilege escalation. Several works have been presented proposing techniques to enhance serverless security monitoring. Serverless functions are ephemeral in nature, making it pretty challenging to use classic approaches of monitoring as serverless functions are spinning up and shutting down based on demand. For this purpose, researchers came up with techniques for continuous runtime monitoring to detect and correct security vulnerabilities in real-time [10]. According to the literature, encryption also presents a very significant aspect of serverless security. Developers need to ensure in-transit and at-rest encryption for sensitive data, especially when functions interact with any external data sources or APIs. Integration of encryption in serverless workflows ensures that in the event of a breach, such data will remain secure. In brief, literature in the serverless domain demands a multi-layered approach with function-level vulnerability, API security, managing permissions and practices for monitoring. Risks associated with serverless computing evolve with its advancement; therefore, it is quite important that developers and organizations know novel threats and the mitigation techniques that come along with this change.

3. Methodology

In this study, we take a mixed-methods approach that discusses both the difficulties and best practices concerning serverless applications regarding security and FaaS. We divided our methodology into three main phases: literature review, case study analysis, and testing using empirical evidence. In the first phase, we performed an extensive literature review of the existing work related to serverless security, primarily drawing from academic papers, industry reports, and other relevant whitepapers within the context of security. This identifies the critical security issues relating to serverless computing and, in particular, FaaS architecture. We analyzed real-life case studies of serverless applications in both enterprise and small- to medium-sized business contexts in order to determine common vulnerabilities. These include API misuse, function-level privilege escalation, and misconfigured permissions. We entered the empiric phase and set up a controlled environment where we could experiment with serverless architectures, particularly through FaaS models. Thus, in the phases of security assessment, employing penetration testing along with runtime monitoring, we checked on known vulnerabilities such as injection attacks and unauthorized access, as well as privilege escalation. Multi-layered security approach, along with all best practices, encryption, least privilege, and API gateways, was applied to our application. We then assessed the impact of these mitigation strategies against the security risks through the measurement of attack vectors, performance impacts, and overall resilience of the system. Quantitative measurements from the performance tests were analyzed through

statistical methods. At the same time, the qualitative views from the case studies provided much-needed contextual information about the security risks and the mitigation strategies. The output of this approach will enable the reader to understand how best practices are implemented to strengthen serverless applications against modern types of security threats.

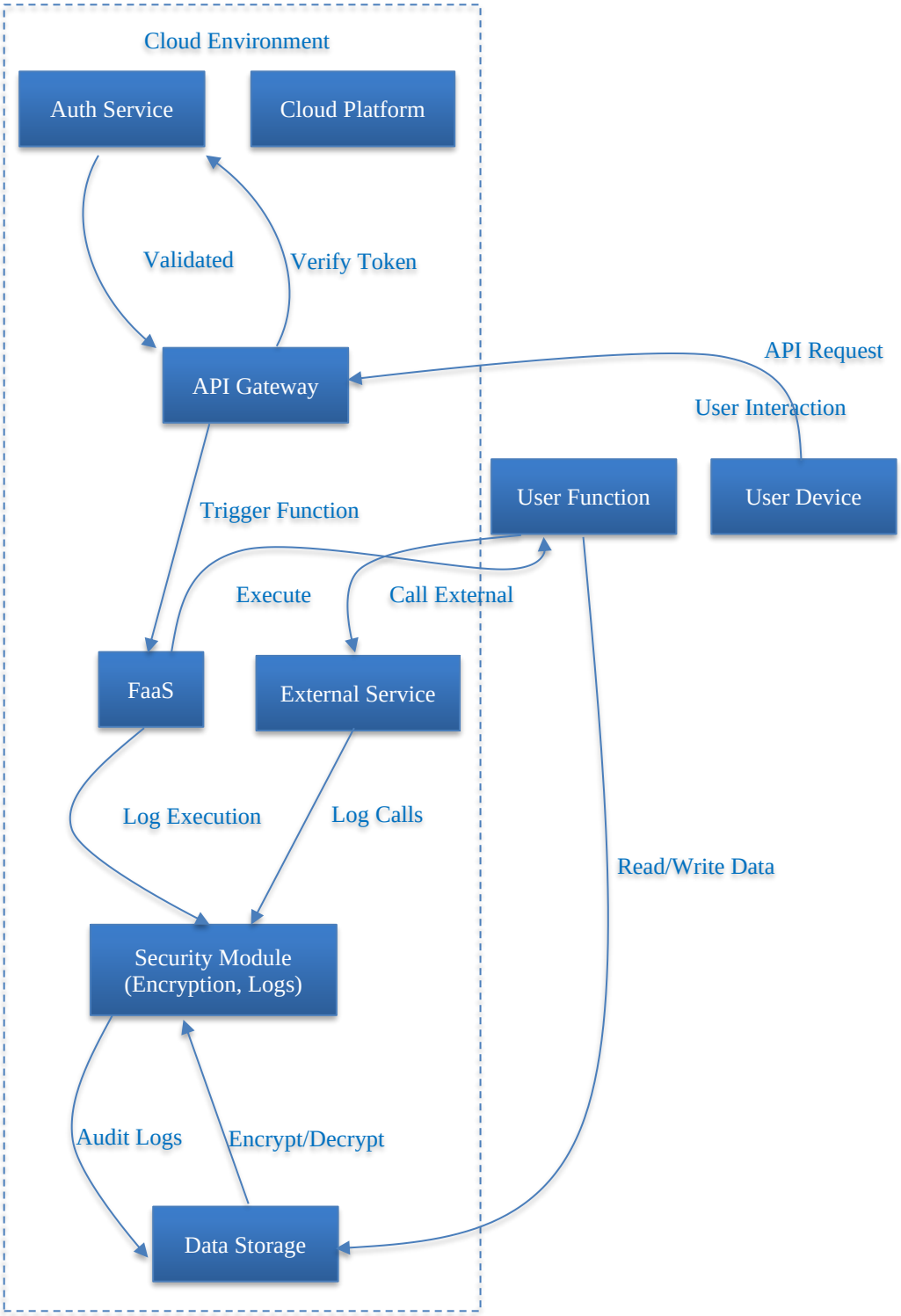

Fig. 1 Simplified Cloud-Based system architecture with secure data processing

In Figure 1, a User Device sends an API request to an API Gateway, authenticated through Auth Service here. Once authenticated, the API Gateway invokes a function in the FaaS module, which is then processed by a User Function. This operation can reach Data Storage to read or write data or invoke an External Service. All data operations are secured and logged through the Security Module, which encrypts, decrypts, and logs all activity for audit logging. Both the FaaS and the External Service log all activities through the Security Module for monitoring and compliance purposes. The architecture thus splits into two subgroups, User Interaction and Cloud Environment, organizing the components for clearer visualization of the process flow within the cloud ecosystem.

4. Data Description

The data used within this study come from various sources, such as public cloud providers, industry reports, and security benchmarks that were provided by the Open Web Application Security Project (OWASP) and the Cloud Security Alliance (CSA). Most empirical data hinged on vulnerabilities at the function levels in FaaS architecture, such as injection attacks, improper access controls, and privilege escalations. The case studies were based on actual implementations of serverless architectures in small-to-medium businesses and large enterprise organizations. Data related to penetration testing were also collected using security testing tools such as OWASP ZAP and Burp Suite to understand possible vulnerabilities in deployed serverless functions.

5. Results

Results from security testing of FaaS architectures provided numerous points in gaining insight into the presence of vulnerabilities and the effectiveness of implemented security best practices. During the penetration testing, several security vulnerabilities were detected during this phase, especially in API security privilege escalation attacks at the function level. Injection attacks represented a very common case of attacking unsecured APIs, agreeing with the fact that poor API management remains one of the greatest threats to serverless applications. Attack Frequency Reduction Rate ⟨AFRR) is:

$$\text{AFRR} = \left(\frac{F_{before} - F_{after}}{F_{before}}\right) \times 100 \quad (1)$$

Where:

F_{before} = Frequency of attacks before security implementation

F_{after} = Frequency of attacks after security implementation

(1) calculates the percentage reduction in attack frequency after implementing security measures. System Latency Increase (SLI) is given as:

$$\text{SLI} = \left(\frac{L_{with\_security} - L_{without\_security}}{L_{without\_security}}\right) \times 100 \quad (2)$$

Where:

$L_{with\_security}$ =Latency with security measures (ms)

$L_{without\_security}$ = Latency without security measures (ms)

(2) calculates the percentage increase in system latency due to the introduction of security measures. Overall Risk Score ⟨ORS) is:

$$\text{ORS} = \sum_{i=1} \quad I_i \times P_i \quad (3)$$

Where:

I_i = impact of the i-th vulnerability

P_i = Probability of the i-th vulnerability occurring

n = number of vulnerabilities

(3) calculates an overall risk score based on the individual impact of vulnerabilities and their occurrence probability. Cost-Benefit Ratio of Security ⟨CBRS) is:

$$\text{CBRS} = \frac{AFRR}{SLI} \quad (4)$$

Where:

AFRR =Attack Frequency Reduction Rate (calculated from the first equation) SLI =System Latency lncrease (calculated from the second equation). (4) evaluates the cost-benefit ratio of implementing security measures in terms of attack frequency reduction and performance impact. Secondly, improper permissions configuration of the functions actually permits privilege escalation attacks; hence, attackers can gain unauthorized access to sensitive data. However, the frequency of successful attacks is managed to be significantly reduced after implementing security best practices like the principle of least privilege and secure API gateways. Input validation mechanisms are used to mitigate the risk of an injection attack, and API security is improved by implementing and using mechanisms for input validation. Functions were also given the minimum necessary permissions with reduced attack surfaces for privilege escalation. Regarding runtime security, one would appreciate the fact that there were more continuous monitoring and logging tools that provided real-time anomaly detection- a sudden spike in traffic, an attempt to access the system without authority, etc.

All of this was proactively monitored, and the threats had potential in case of security breaches; it could never help the threats escalate. Secondly, the encryption mechanism on the data in transit and the one resting added an extra layer of security. They ensured that even though one intends to leak the information, the given information will not allow unauthorized access. In addition, any performance analysis revealed that the enforcement of those mechanisms didn't degrade the system performance drastically. However, the encryption and monitoring effects introduced some minor latency. Yet, the added security achieved was tolerable since, especially in enterprise environments, security matters are considered. Adopting a multi-layered approach to security helped mitigate vulnerabilities without diminishing the

overall functionality and responsiveness of these serverless applications. The results pointed out that the design and deployment of FaaS-based applications must stay rigid on security best practices. Therefore, concentration on function-level security, as well as greater system-wide precautions, would dramatically decrease the risk of breach occurrence in this particular serverless environment.

Table 1. Vulnerabilities typically found in FaaS Architecture

Vulnerability	Impact Level	Attack Vector
Injection Attacks	High	Exploits insufficient input validation in APIs
Privilege Escalation	Critical	Escalates permissions through misconfigured IAM roles
Unauthorized Access	High	Gains access through weak authentication methods
Insecure API	Medium	API without proper validation and security measures
Data Leakage	Critical	Improper encryption of data in transit or at rest

Table 1 enumerates a few common vulnerabilities associated with FaaS architectures. Five key vulnerabilities have been identified: injection attack, privilege escalation, unauthorized access, insecure API, and data leakage. Every vulnerability level ranges from medium to critical impact level. The table also provides the exact attack vector for every vulnerability- including the lack of input validation on the input side for injection attacks and weak encryption for data leakage. The cases of critical privilege escalations and data leakage highlight proper access control in serverless environments and appropriate encryption. This table highlights focal points for improvement of security in FaaS applications.

Table 2. Security best practices with effectiveness

Security Best Practice	Effectiveness (Reduced Attack Surface)	Description
Encryption	High	Encrypts sensitive data to prevent exposure
Least Privilege Access	Critical	Limits functions to minimum permissions required
API Validation	High	Validates inputs to prevent injection attacks
Runtime Monitoring	Medium	Monitors function behavior to detect anomalies
IAM Policy Configuration	Critical	Configures access policies to prevent privilege escalation

Table 2 presents the five best security practices towards reducing the risks in FaaS applications encryption, least privilege access, API validation, runtime monitoring, and IAM policy configuration. Their effectiveness ranks on the level of reducing the attack surface from a medium to a critical level. For each one of the security best practices, I give a description of the function performed by it. For example, encryption prevents access to sensitive information, and least privilege access prevents escalations in permissions. This table presents recommendations for security prioritization that enhances the holistic security posture of a serverless system.

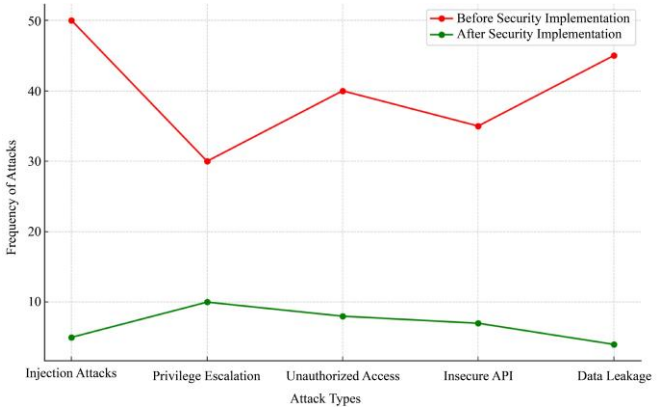

Fig. 2 Attack frequency before and after security implementations

Fig 2 illustrates the different types of attacks and their different frequencies both before and after the implementation of the security measures in serverless FaaS architectures. Before the deployment of the security measures, attacks like injection attacks, privilege escalation, and data leakage frequency were extremely high. However, when security best practices like API validation, least privilege, and encryption are put into place, the frequency of the attack decreases dramatically. The graph above indicates the effectiveness of security as both types of attacks are shown to drastically decrease; however, injection attacks decreased from 50 incidences to merely 5. Such a drop shows that having proactive security strategies in serverless environments is very valuable.

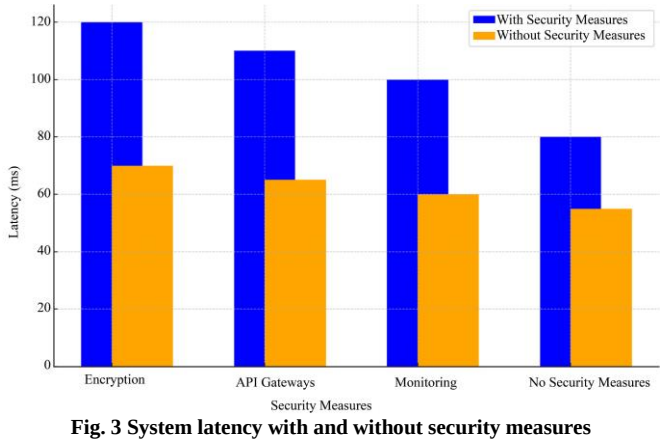

Fig. 3 System latency with and without security measures

Figure 3. System latency without and with security: encryption, API gateways, and monitoring As shown in Figure 3, the system latency with security is increased; however, the increase is very small, which suggests that the performance/security trade-off is acceptable. For instance, using security measures like encryption incurs a latency overhead on the order of 120 ms as compared to 70 ms without security; the additional protection against attacks renders this latency acceptable. The graph further shows that robust security does not come at the expense of dramatic loss in system performance in serverless applications.

6. Discussions

The data presented in this paper, in tables and figures, explains crucial findings on vulnerabilities in FaaS architectures and the effectiveness of security measures to reduce these risks. There are five key critical vulnerabilities that are commonly found in FaaS environments, as identified in Table 1, such as injection attacks, privilege escalation, unauthorized access, insecure APIs, and leakage. A practical example of these security best practices can be seen in the implementation of a Secure Serverless Claims Processing System in the healthcare insurance industry. Healthcare insurance companies manage sensitive personal and medical information during claims processing, making security a critical concern. Traditional infrastructure is often slow, prone to misconfigurations, and expensive to scale. Leveraging a serverless FaaS architecture enables healthcare insurers to process claims in a more secure, scalable, and cost-effective manner while adhering to regulatory standards like HIPAA. The system employs an API Gateway to handle requests securely and serverless functions to manage each step of the claims lifecycle, such as intake, validation, fraud detection, adjudication, and notifications. Sensitive data, including personal health information (PHI), is encrypted both in transit and at rest to ensure compliance. Least privilege access is enforced to minimize risk, while APIs are secured with input validation mechanisms and OAuth tokens. Real-time monitoring detects any anomalies, and audit logs are encrypted for compliance purposes. This serverless model supports scalability, cost efficiency, and high-level security, protecting sensitive data while ensuring regulatory compliance. The tabulation emphasizes how severe these vulnerabilities are since the impact levels range from medium to critical. Injection attacks, for instance, emanate from the absence of proper input validation and leave FaaS functions appreciably vulnerable to code injections. Privilege escalation and data leakage are two major vulnerabilities rated critical that require better controls in terms of access and stronger encryption practices. For instance, there is an emphasis on vulnerable points in serverless applications due to security measures influencing them. A corresponding set of security best practices is provided in Table 2, proving the ability to shrink the attack surface. At the high level of effectiveness are those related to encryption and least privilege access, which stand out as being highly advisable, while those associated with API validation, runtime monitoring, and IAM policy configuration come out at medium to high. This emphasizes the multi-layered approach as the sound methodology that is going to secure FaaS architectures, where encryption maintains data integrity and least privilege access minimizes the possibility of privilege escalation by limiting permissions only to what is required for the functioning process of functions. API validation has been very crucial in preventing injection attacks since insecure APIs are among the most common vulnerabilities in the serverless environment.

Figure 2 illustrates how often attacks occur before and after a security feature or combination of features is added, further depicting the criticality of the above security features. Injection attacks, escalation, and unauthorized access are all common attacks that frequently occur in the absence of security measures, which in some cases reached a frequency of up to 50. However, when such best practices as API validation, least privilege access, and encryption were added, such attacks saw their frequency drop enormously. For example, injection attacks dropped to just 5 and privilege escalation reduced to 10, making for a pretty big improvement in the security posture of the system. This points out that following best practices improves resilience against common attack vectors fairly significantly for serverless applications. It also relates to how proactive security measures with regard to minimizing access and securing data flows are particularly fundamental in a serverless context where functions are stateless and frequently interact with third-party services. Operational trade-offs the application of FaaS security introduces are further insights from Figure 3, where system latency with and without security measures has been plotted. Intuitively, at least, the presence of security measures encryption to API gateways and monitoring adds to system latency. Evidently, latency due to encryption is at 120 ms compared with 70 ms when no security measure is added. However, it is not very significant given added security, and the system still performs well within acceptance limits, especially for sensitive applications wherein security will not be compromised. Thus, it can be assumed that security may incur some overhead operationally but does not significantly impact the performance of serverless applications. The marginal increase in latency is an acceptable trade-off for the significant decrease in vulnerability to attacks. This further enforces the broad principle of balancing security with performance as a crucial determinant of which businesses will deploy FaaS applications at scale.

In summary, the data present clear proof that serverless FaaS architectures are not only highly efficient and scalable but also vulnerable to certain critical security threats that need a layered security approach. Results from Tables 1 and 2 and Figures 2 and 3 present the importance of security best practices that may include encryption, API validation, and least privilege access, which can considerably reduce the chances of attack. It also shows that, despite slight operational

overheads by security implementations, the improvements in security far outweigh the costs. Thus, this supports the argument for incorporating the principles of security-first into the design of serverless applications, making them both very functional and safe against dynamic threats.

7. Conclusion

Coming from the analysis of vulnerabilities and security practices in the architectures of FaaS it emerges with a big critical need for a strong multi-layered security approach. From the data from Table 1, the following present significant vulnerabilities: injection attacks, privilege escalation, and insecure APIs, which pose a considerable threat to serverless environments. However, as indicated in Table 2 and Figure 2, security best practices in the form of encryption, least privilege access, API validation, and runtime monitoring show much promise in effectively mitigating these risks. The dramatic reductions in attack frequency following the application of these measures are indicative of their success. Further, security-performance trade-offs, as seen in Figure 3, demonstrate that though adding latency, overhead from encryption and API gateways are pretty low as opposed to tremendous gains in the security of the system. Such outcomes highlight security as an important concern to be addressed at the design and deployment time of FaaS-based applications. Organizations can greatly reduce their serverless environment's attack surface while maintaining operational efficiency by embracing best practices as part of an active security strategy. The ultimate key for companies to enjoy the servers' benefits of computing without compromising either the integrity of data or the resilience of the system is securing FaaS applications.

Limitations

Although it advances the body of knowledge on the security challenges and best practices of FaaS, this study certainly has some limitations. To start with, the empirical testing carried out is one of the simulated environments, which may not reflect the complexity of real-world serverless deployments. Although controlled experiments are conducted, the tests may not pick up on all the subtler security issues that arise in large-scale, production-level applications involving dynamic variables such as traffic loads and multi-tenant environments, which can introduce their own set of risks. Another limitation in scope is the scope of security best practices implemented. While this paper merely concentrated on some essential practices such as encryption, least privilege access, and API security, the greater security measures were yet not addressed: other particular compliance requirements for serverless, say, and also more focused automated patch management.

Future Scope

With the adoption of serverless computing on the rise, security challenges continuously evolve and require continuous innovation and research. In order to further enhance studies in this area, several key domains need further study to provide a proper understanding of serverless security. The first area is doing more in-depth studies to explore real-world, scale-deployed serverless applications. Such studies would provide valuable insights into unique security concerns that arise in multi-tenant, production-level environments, traffic patterns, shared resources, and third-party dependencies, which adds complexity to security management. As cloud providers continue to innovate using new tools and technologies tailored to serverless architectures, they will keep emerging. This future work should research how inventions in machine learning, self-healing systems, and automatic threat detection could be combined with serverless environments to provide better security monitoring and improved response times. Another area of investigation for future research will be the development of standards and regulatory frameworks specific to serverless. With the prevalence of serverless in industries that handle sensitive data, such as healthcare and finance, organizations will have to pay heightened attention to ensuring that their serverless deployments comply with evolving legal and regulatory requirements. Finally, future research should explore the long-term cost implications of implementing security in serverless architectures. This would provide an opportunity for direct researchers to analyze and make trade-offs between security measures and operational efficiency so that they can help organizations find the right balance between security and performance for their serverless applications.

References

[1] Gabriele Russo Russo, Valeria Cardellini, and Francesco Lo Presti, "Serverless Functions in the Cloud-Edge Continuum: Challenges and Opportunities," *2023 31st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, Naples, Italy, pp. 321-328, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[2] Xing Li, Xue Leng, and Yan Chen, "Securing Serverless Computing: Challenges, Solutions, and Opportunities," *IEEE Network*, vol. 37, no. 2, pp. 166-173, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[3] Bader Alouffi et al., "A Systematic Literature Review on Cloud Computing Security: Threats and Mitigation Strategies," *IEEE Access*, vol. 9, pp. 57792-57807, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[4] Sanjaa Bold, and Batchimeg Sosorbaram, "Security and Privacy Concerns of the Internet of Things? (IoT) in IT and its Help in the Various Sectors across the World," *International Journal of Computer Trends and Technology*, vol. 68, no. 4, pp. 266-272, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[5] Marwa A. Elsayed, and Mohammad Zulkernine "PredictDeep: Security Analytics as a Service for Anomaly Detection and Prediction," *IEEE Access*, vol. 8, pp. 45184-45197, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[6] Ben Wang et al., "Security-Aware Service Function Chaining and Embedding with Asymmetric Dedicated Protection," *IEEE Access*, vol. 12, pp. 53944-53957, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[7] Qinzhe Wu, and Lizy K. John, "Performance of Java in Function-as-a-Service Computing," *Proceeding 2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC)*, Vancouver, WA, USA, pp. 261-266, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Hassan B. Hassan, Saman A. Barakat, and Qusay I. Sarhan, "Survey on Serverless Computing," *Journal of Cloud* Computing, vol. 10, pp. 1-29, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[9] John Michener, "Security Issues with Functions as a Service," *IT Professional*, vol. 22, no. 5, pp. 24-31, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[10] Eduard Marin, Diego Perino, and Roberto Di Pietro "Serverless Computing: A Security Perspective," *Journal of Cloud Computing*, vol. 11, no. 1, pp. 1-12, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[11] Johannes Manner et al., "Cold Start Influencing Factors in Function as a Service," *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, Zurich, Switzerland, pp. 181-188, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[12] Paweł Żuk, and Krzysztof Rzadca, "Scheduling Methods to Reduce Response Latency of Function as a Service," *2020 IEEE 32nd International Symposium on Computer Architecture and High-Performance Computing (SBAC-PAD)*, Porto, Portugal, pp. 132-140, 2020. [CrossRef] [Google Scholar] [Publisher Link]