DOI: 10.2507/35th.daaam.proceedings.041

SERVERLESS ARCHITECTURE AND SECURITY

Marko Harambasa, Karlo Josic & Matej Basic*

This Publication has to be referred as: Harambasa, M[arko]; Josic, K[arlo] & Basic, M[atej] (2024). Serverless Architecture and Security, Proceedings of the 35th DAAAM International Symposium, pp.0299-0305, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-44-0, ISSN 1726-9679, Vienna, Austria
DOI: 10.2507/35th.daaam.proceedings.041

Abstract

This paper explores the serverless architecture, detailing its key components, benefits, and security concerns. It contrasts two main types: Function as a Service (FaaS) and Backend as a Service (BaaS), highlighting how they allow for building applications without the need for managing underlying servers. The paper continues into serverless architecture's operational and financial benefits, such as scalability, reduced maintenance, and cost efficiency. Security challenges unique to serverless computing are examined, stressing the importance of adopting robust security measures like comprehensive monitoring and logging to mitigate risks. The paper discusses the inherent security benefits of serverless computing, including minimised attack surfaces and automated updates. It outlines best practices for ensuring secure serverless environments, such as secure coding, stringent access controls, data encryption, and continuous monitoring.

Keywords: Serverless; Function as a Service (FaaS); Backend as a Service (BaaS), network security.

1. Introduction

In the last few years, serverless architecture has become the revolutionary approach in cloud computing, providing a way to build and deploy applications and services without the need to know and configure the underlying infrastructure. The heart of the serverless architecture is deploying the code in response to events, which means the application is running only when needed, eliminating the need for traditional server management. Two key concepts are Function as a Service (FaaS) and Backend as a Service (BaaS). FaaS focuses on executing code in response to some action or events, while BaaS provides automated backend services [1].

These two approaches speed up the development life cycle, providing quicker feature release and application updates. The serverless architecture can be used with today's three most extensive serverless technologies: AWS Lambda, Azure Functions, and Google Cloud Functions. Those three serverless technologies are also called functions since the serverless architecture combines multiple small functions. All other resources on the internet that we are using have some pros and cons, some of which are serverless. This paper will cover all the excellent and tricky parts of serverless architecture, such as scalability and cost-efficiency, and critically examine the security landscape unique to serverless environments [2]. Serverless architecture inherently enhances development agility by abstracting the complexity of the underlying infrastructure, allowing developers to focus more on coding than operational concerns. However, it also introduces challenges in monitoring and debugging because traditional tools are often not equipped to handle the ephemeral nature of serverless components.

The paper aims to provide a comprehensive examination of serverless architecture, focusing on its advantages and challenges. In the paper, we will present the benefits of serverless architecture, including its scalability, cost-efficiency, and ability to speed up the development life cycle and compare the most extensive serverless technologies: AWS Lambda, Azure Functions, and Google Cloud Functions.

2. Core Components of Serverless Architecture

Serverless architecture is a cloud computing model where the cloud service provider dynamically manages the allocation of machine resources. In a serverless architecture, developers focus on writing code without worrying about the underlying infrastructure, such as servers or virtual machines [3]. Figure 1 shows the Serverless Architecture diagram from a high-level design view.

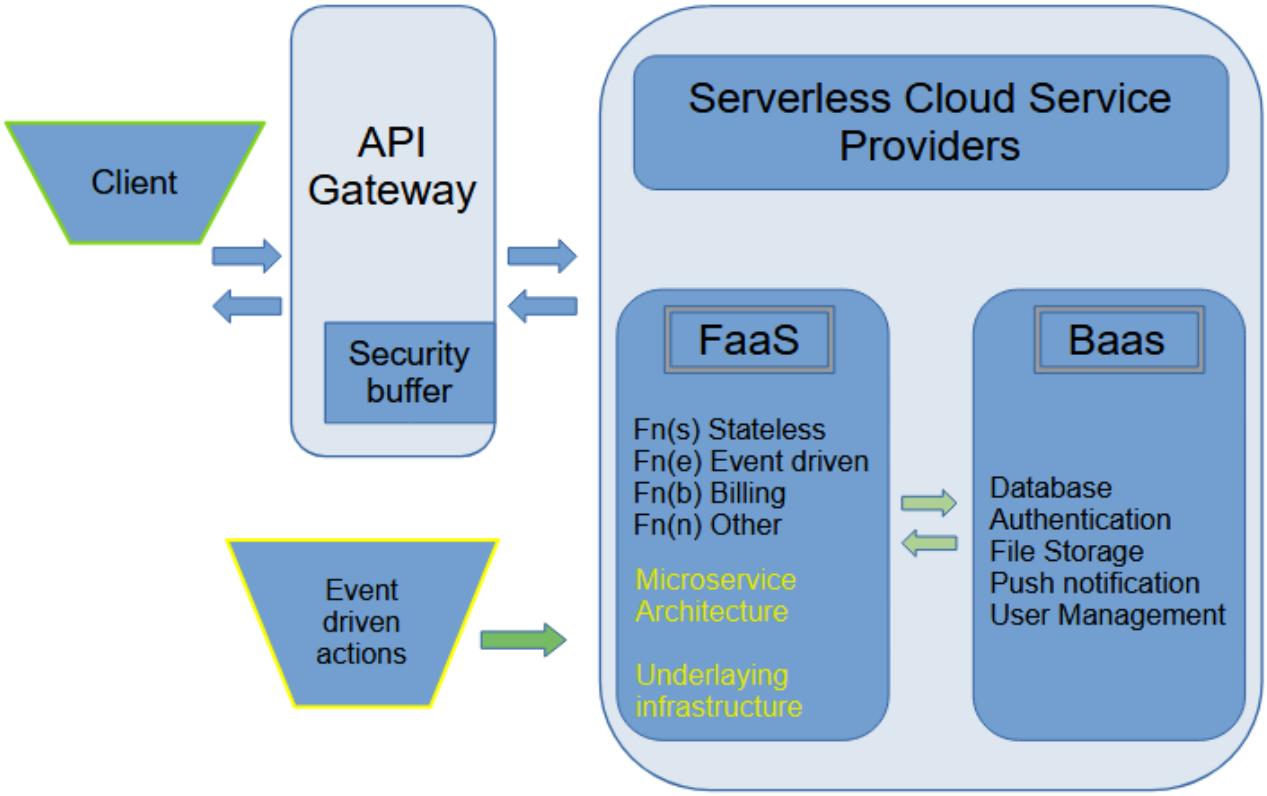

Fig. 1. Diagram of Serverless Architecture

2.1. Function as a Service (FaaS)

"FaaS" is the heart of the serverless architecture, allowing developers to write and deploy code executed in response to events without the complexities of managing the underlying infrastructure. Focusing on the individual functions and excludes servers entirely. There are a few steps to how "FaaS" works:

- Event-Driven Execute: functions are invoked by specific events, for example, HTTP requests, file uploads, or database operations
- Stateless Functions: each function call is treated as an independent event, with no shared state between executions
- Scalability and Auto-Scaling: as the demand or workload increases, more instances of the function are automatically created to handle the load
- Micro billing: billing is based on the actual usage, typically calculated as a combination of the execution time and the number of times the function is triggered.

2.2. Backend as a Service (Baas)

Backend as a Service (BaaS) is a cloud computing model that provides backend services and functionalities to developers without requiring them to manage the underlying infrastructure. In the context of serverless architecture, BaaS offers pre-built backend services, such as databases, authentication, file storage, push notifications, and user management, which developers can easily integrate into their applications.

The roles the "BaaS" involved in the serverless architecture:

- Managed Backend Services: The cloud provider manages several backend services BaaS offers, including database administration, user authentication, cloud storage, and server-side logic
- Emphasis on Frontend Development: Since the service provider handles the backend of apps when using BaaS, developers may concentrate more on creating the front end of their projects
- Integration with FaaS: BaaS and FaaS are frequently combined to provide a complete serverless architecture, with BaaS managing standard backend operations and FaaS handling business logic in response to events.

2.3. Key Players and Technologies

Several major companies dominate the serverless market and provide reliable FaaS and BaaS solutions:

- One of the first offerings is AWS Lambda (Amazon Web Services), which enables running code without the need to provision or manage servers. Its seamless integration with other AWS services enables a serverless architecture
- Microsoft Azure's Azure Functions: Provides event-driven serverless computing that seamlessly integrates into the more extensive Azure network of services
- Google Cloud Functions: A fully managed environment within the Google Cloud Platform runs code in response to events. It is a scalable, pay-as-you-go function as a service solution [4].

3. Benefits of Serverless Architecture

Serverless architecture presents many advantages, such as minimised operational burdens, enhanced cost-effectiveness, scalability, and expedited time-to-market. It is a compelling option for numerous organisations developing contemporary and scalable applications.

3.1. Cost Efficiency in Serverless Architecture

Cost efficiency is the most valuable thing when implementing serverless architecture. That stood out from the unique billing model and reduced operational costs. It allows us to optimise the price of the architecture at the minimum level. Nowadays, there are different types of cost models for serverless architecture:

- Pay-Per-Use Billing Model: Unlike traditional architectures, where resources are paid for regardless of use, serverless computing follows a pay-per-use model. Costs are incurred based on the number of executions and the duration of the code execution, down to the millisecond. This granular billing approach means no charge if a function is not running.
- Reduced Infrastructure Costs: Since the cloud provider manages the infrastructure, organisations save on the costs associated with provisioning, maintaining, and scaling physical servers or virtual machines
- Decrease in Over-Provisioning: Serverless computing eliminates the need for guesswork in capacity planning [5].

3.2. Scalability and Performance

After cost considerations, scalability and performance are the primary focal points in an application's development. The application must function consistently, whether accessed by a single user or thousands. In serverless architecture, there is a couple of stuff that is different from traditional architecture, such as:

- Automatic Scalability: Serverless functions automatically scale depending on the workload. This means that during periods of high demand, the architecture can scale up seamlessly to handle the increase in requests and scale down during low-demand periods, ensuring efficient resource utilisation.
- Performance Optimization: The underlying serverless computing infrastructure is optimised for performance. Cloud providers continually upgrade and maintain their systems, ensuring serverless functions run on high-performance computing resources.
- Instantaneous Scaling: The ability to instantly scale up or down in response to incoming requests ensures that serverless applications can maintain performance levels regardless of the number of requests they receive.

3.3. Maintenance and Operational Efficiency

Maintenance and Operations are necessary for ongoing server upkeep and slashing expenses linked with hosting applications. This technology empowers businesses to trim server maintenance costs, enhancing efficiency and scalability. While serverless computing eliminates hardware expenses tied to conventional web servers, there might still be some operational overhead related to overseeing serverless applications, such as configuring logging and monitoring metrics from various origins. Over recent years, serverless computing has become a preferred solution for bespoke software developers. It reduces maintenance and improves operation efficiency in server ways, such as:

- Reduced Maintenance Overhead: With the server management and operational responsibilities shifted to the cloud provider, developers and IT teams are freed from routine maintenance tasks such as patching, updating, and managing server health
- Focus on Core Product Development: Thanks to this infrastructure management offloading, organisations may refocus their attention and resources on core product development and innovation instead of being mired in operational issues.

4. Security in Serverless Architecture

Since serverless users work with cloud providers, they shift their security responsibility to them. In deploying serverless applications, we keep control over most of the stack to our cloud provider, and they provide services such as key management. No longer own OS hardening, admin rights, SSH, and segmentation. Cloud providers are reliable in keeping their parts of the stack patched and secured, so giving them a more significant portion of the stack certainly improves things. Additionally, the temporary, stateless nature of serverless computing makes the attacker's liver harder. Serverless functions like AWS Lambda run for a few seconds and then die, and containers get recycled. The faster serverless function comes and goes, the less memory is required, and having no memory reduces the risk of long-term attacks. The fact that serverless applications are structured as a more significant number of small functions in the cloud provides a fantastic opportunity for security. Application security tools often go to incredible lengths to analyse and instrument our packaged application to observe or filter the internal flow of applications. Moving to smaller microservices enables more fine-grained IAMs. The opportunity to apply security policies to each of those small things can significantly reduce the attack surface. In serverless architecture, security assumes a different dimension than traditional server-based setups. While the cloud provider manages the infrastructure's security, application-level security, particularly the code and data, essentially remains the developer's responsibility.

Security in serverless architecture encompasses various crucial topics to protect data, applications, and resources. Some critical issues include:

- **Shift in Security Responsibility**: The serverless model shifts specific security responsibilities to the cloud service provider. However, this means only some security concerns are outsourced. Issues like application vulnerabilities, data encryption, and access controls still need vigilant management.
- **Unique Security Context:** Serverless functions are stateless and ephemeral, meaning they do not traditionally persist. Their transient nature requires a different approach to monitoring and securing them than persistent server environments [6].

4.1. Common Security Challenges

In serverless architecture, several common security challenges must be addressed to ensure the robust protection of applications and data. Here are some of these challenges:

- **Third-party Dependencies:** Third-party libraries and APIs are frequently used by serverless applications. Each external component introduces potential vulnerabilities. It is essential to make sure these dependencies are current and safe
- **Insecure Serverless Deployment Configurations:** Security vulnerabilities in serverless apps might arise from deployment errors. These include failing to establish appropriate authentication and permission procedures, having too liberal access rights, and not keeping enough logs.
- **Vulnerabilities in Application Code:** Since serverless computing strongly emphasises function execution, security breaches may result from any weakness in the code, including injection errors or incorrect exception handling
- **Limited Monitoring and Visibility:** Standard monitoring solutions might also not work in a serverless system. The monitoring services the cloud provider provides where the serverless is hosted can enable the logging application.
- **Risks of Denial of Service (DoS):** Even if serverless architecture can handle heavy loads, DoS attacks can still occur.

These attacks could cause many function executions, resulting in significant costs and possible service degradation.

4.2. Serverless Security Threats

While attackers may still leverage similar motivations, they will employ different tactics when targeting serverless applications, as these applications are architecturally different from those we're used to. Below are the specific security risks inherent to this novel application architecture.

4.2.1. The Threat of Over-Privileged Functions

With serverless applications, applying permissions to specific functions and ensuring those privileges are limited to the bare minimum is possible. It will be able to lessen the impact of any strike and drastically reduce the attack surface by doing this. Regretfully, a recent Check Point study revealed [7] that few developers utilise this possibility.

According to analysis, 16% of serverless application tasks are deemed “serious,” and 98% of functions are in danger. Additionally, to increase the security of the function and the application, most of these functions could have their permissions granted less than necessary. Check Point rates the risk associated with each function it analyses. That is based on the identified posture flaws, considering the type of weakness and the environment in which it manifests. Tens of thousands of functions in live applications were scanned, and the results showed that most serverless apps are just not implemented as securely as they should be to reduce risks, something that OWASP (The Open Worldwide Application Security Project) controls mention more than a few times in their top 10 list. Check Point found that unnecessary permissions are the main security posture problems; the remaining issues are insecure code and setups.

4.2.2. The Groundhog Day Attack

Attackers find it harder to stay in applications over time since serverless functions are transient and fleeting. Attackers will still carry out attacks; their tactics will vary, making their lives more difficult. Because serverless functions have a short lifespan, serverless security risks could evolve. An attacker might create a far shorter attack that only takes a little information, such as a few credit card numbers. What is known as the "Groundhog Day" attack is a single attack round that keeps repeating itself.

4.2.3. Poisoning the Well

Despite the short lifespan of cloud-native resources, attackers can still establish long-term application persistence. They achieve this by exploiting a technique known as "Poisoning the Well," which circumvents the transient nature of serverless apps. Cloud-native applications often comprise multiple modules and libraries with code sourced from various third-party providers. Attackers aim to insert malicious code into these applications. Subsequently, the malicious code can communicate with its source, receive instructions, and wreak havoc after poisoning the well [8].

5. Security Best Practices for Serverless Architecture

Like any other technology, serverless architecture is not immune to vulnerabilities. Therefore, it's crucial to minimise risks as much as possible. Implementing best practices for serverless architecture, also recommended by cloud providers, is essential in achieving this goal.

5.1. Use API Gateways as Security Buffers

The most common protocol for communication between the client and server is HTTP protocol. The function shouldn’t be exposed directly to the front end; it must be wrapped using the API gateway service. Using API HTTPS endpoint gateways to divide data from functions is one technique to stop event-data injection in serverless apps. An API gateway will serve as a security buffer when data is collected from various sources, separating app users on the client side from serverless functions on the backend. Implementing a reverse proxy to provide several security checks lowers the attack surface. Built-in security measures, such as data encryption and key management provided by the provider, can be used by HTTP endpoints. These measures help to protect sensitive data, environment variables, and stored data.

5.2. Data Separation and Secure Configurations

Implementing preventive measures such as code scanning, isolating commands and queries, identifying any exposed secret keys or unlinked triggers, and configuring them according to the recommended practices by the CSP's serverless application are vital strategies to thwart DoS attacks. Additionally, to mitigate the risk of DoS attacks disrupting execution calls, setting function timeouts as low as possible is advisable.

5.3. Dealing with Insecure Authentication

Implementing various specialised access control and authentication services is crucial to minimise the likelihood of authentication failures. Leveraging the CSP's access control solutions, such as OAuth, OIDC, SAML, OpenID Connect, and multi-factor authentication (MFA), can enhance the complexity of authentication, making it more difficult to compromise. Additionally, imposing unique password complexity requirements, including character type and length constraints, impedes hackers' attempts to crack passwords.

5.4. Sufficient Serverless Monitoring and Logging

Monitoring and logging provide users and operators with a comprehensive view of tracking the behaviour of our functions and workflow. Implementing monitoring and logging is essential to gain insights into every serverless application operation.

Relying solely on the CSP's logging and monitoring capabilities is inadequate as they do not encompass the application layer. It's also imperative to consider what data is being logged, ensuring that sensitive information that should remain undisclosed is not logged, as it could serve as an entry point for attackers.

5.5. Minimize Privileges

They diverge the functions from each other and limit their rules by provisioning IAM roles based on their rights. Minimising the privileges in independent functions is crucial. Another way to do this would be to make sure the code executes with the fewest permissions necessary to act.

5.6. Separate Application Development Environments

One of the most significant development practices is maintaining continuous development, integration, and deployment (CI/CD) by keeping the different environments apart from staging, development, and production [9]. This guarantees that appropriate vulnerability management is prioritised at every development stage before pushing that code version forward. Additionally, it ensures that the program is continuously tested and improved by prioritising patches, protecting updates, and detecting vulnerabilities, all of which help developers stay one step ahead of attackers.

6. Test case example

Netflix, a streaming service with 125 million customers, delivers 10 billion hours of video content each quarter. This feat is accomplished mainly through its infrastructure, which is built predominantly on Amazon Web Services (AWS). With hundreds of thousands of files and petabytes of data, scaling the application becomes imperative. The service reaches millions of users across 55 countries, providing regularly updated content. Over seven years, Netflix underwent a significant strategic transition, fully migrating to AWS cloud infrastructure. Leveraging serverless technology, particularly AWS Lambda, has become integral to many aspects of its operations. For instance, Lambda functions are vital in processing the vast number of video clips submitted daily, dividing them into segments, and encoding them into the various streams required for Netflix's functionality. Additionally, Lambdas are employed in backend systems for tasks such as verifying file integrity, creating data backups, and resolving issues with the backup process, given the substantial number of daily file modifications. Lambda is also utilised for security purposes, ensuring that all operations, including instance launches and terminations, adhere to established standards. It actively detects unauthorised access attempts and notifies users accordingly. Furthermore, Netflix utilises monitoring tools provided by AWS to track application traffic, enabling the detection of suspicious behaviour.

By embracing serverless architecture, Netflix effectively delegates server management responsibilities to the provider, ensuring that Lambda manages server deployment, compliance, and configuration. This approach guarantees efficient provisioning processes and rapid adaptation to evolving business requirements. Netflix has demonstrated ingenuity in harnessing serverless technologies, particularly AWS Lambda, to manage extensive data volumes, uphold security and compliance standards, and optimise operational efficiency within its cloud-based streaming service [10].

7. Future research

Serverless computing presents several promising scientific research areas, especially security and architecture. One vital research direction addresses the security challenges unique to serverless models, which heavily depend on external services and infrastructure. Studies could investigate how to adapt or redesign conventional security frameworks to suit better the serverless environment, where the short-lived nature of functions makes traditional security monitoring and incident handling methods less effective. This research might include developing new strategies for detecting and addressing security threats tailored to the fleeting characteristics of serverless components.

Another critical area of research focuses on improving resource management and performance within serverless architectures. The on-demand resource provisioning central to serverless computing introduces issues regarding efficient resource use and the reduction of latency during function initialisation, known as cold starts. Future studies might look into creating algorithms and architectural improvements to predict better and manage resource needs, possibly using artificial intelligence to forecast application demands dynamically. Furthermore, there is potential for exploring how serverless computing could be effectively merged with cutting-edge technologies like edge computing. This exploration could enhance how computational tasks are managed and performed across hybrid architectures, significantly benefiting applications that require immediate data processing, such as those involving the Internet of Things (IoT).

8. Conclusion

Flexibility and efficiency are significant benefits that serverless architecture provides, and application to application can handle enormous amounts of users simultaneously. Also, it speeds up the development lifecycle, providing quicker feature releases and bug fixes.

Moving risks to the cloud provided and having available cloud-supplied support is a significant improvement compared to traditional application management, where we are responsible for the server and the infrastructure. By controlling permissions on Lambdas to specify which services in the cloud can call them and which methods within those services can access them, we gain complete flexibility in managing access to Lambdas. Implementing effective data encryption, secure coding techniques, and robust identification and access controls is imperative. Monitoring and logging play crucial roles in ensuring the security of a serverless system. The scalability and reduced operating costs offered by serverless applications are compelling reasons many organisations adopt this approach. Serverless technology represents the future of development and application execution, serving as an entry point into the microservices architecture, a pivotal architectural paradigm in today's and future technological landscape.

By offloading infrastructure management to the cloud provider, serverless architecture frees up valuable time and resources for organisations to focus on innovation and core business objectives. Developers can concentrate on building innovative features and applications, rather than spending time on infrastructure maintenance tasks. This focus on innovation can drive competitive advantage and differentiation in the marketplace.

9. References

[1] A. Sabbioni, C. Mazzocca, M. Colajanni, R. Montanari & A. Corradi (2022). A Fully Decentralized Architecture for Access Control Verification in Serverless Environments, Proceedings of 2022 IEEE Symposium on Computers and Communications (ISCC), ISBN:978-1-6654-9792-3, ISSN: 2642-7389,pp. 1-6, DOI: 10.1109/iscc55528.2022.9912764

[2] https://www.cloudflare.com/en-gb/learning/serverless/what-is-serverless/ (2013), Cloudflare, Accessed on: 2023-11-16

[3] A. Koschel, S. Klassen, K. Jdiya, M. Schaaf & I. Astrova (2021). Cloud Computing: Serverless, Proceedings of 2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA), ISBN:978-1-6654-0032-9, pp. 1-7 DOI: 10.1109/iisa52424.2021.9555534

[4] https://www.hitechnectar.com/blogs/baas-vs-faas-explaining-the-two-serverless-architectures/ (2023). HiTechNectar Trending IT Analysis & Technology News, Accessed on: 2023-11-16

[5] https://www.educative.io/answers/cost-scalability-and-performance-in-a-serverless-architecture (2023). Educative: Interactive Courses for Software Developers, Accessed on: 2023-11-16

[6] A. Serckumecka, I. Medeiros, and A. Bessani (2019). Low-Cost Serverless SIEM in the Cloud, Proceedings of 38th Symposium on Reliable Distributed Systems (SRDS), ISBN:978-1-7281-4222-7, ISSN: 2575-8462, pp. 381-3811, DOI: 10.1109/srds47363.2019.00057

[7] https://www.checkpoint.com/cyber-hub/cloud-security/what-is-serverless-security/ (2023). Cyber Security Solutions, Check Point Software, Accessed on: 2023-11-16

[8] Moric, Z.; Redzepagic, J. & Gatti, F. (2021). Enterprise Tools for Data Forensics. Annals of DAAAM & Proceedings, 10(2), Published by DAAAM International, ISBN 978-3-902734-33-4, ISSN 1726-9679, Vienna, Austria, DOI: 10.2507/32nd.daaam.proceedings.014

[9] Dakić V.; Redžepagić J.; Bašić M. (2022). CI/CD Toolset Security, 33rd DAAAM International Symposium on Intelligent Manufacturing and Automation, Published by DAAAM International, ISBN 978-3-902734-36-5, ISSN 1726-9679, Vienna, Austria, DOI: 10.2507/33rd.daaam.proceedings.022

[10] https://dashbird.io/blog/serverless-case-study-netflix/ (2023). Dashbird blog, Accessed on: 2023-12-28