World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(RESEARCH ARTICLE)

Check for updates

Threat modelling for serverless architectures identifying and mitigating risks in Function-As-A-Service (FAAS)

Karthik Venkatesh Ratnam [1] and Rajashekar Reddy Yasani [2,*]

[1] *Southern Methodist University. USA.*
[2] *Murray State University. USA.*

World Journal of Advanced Research and Reviews, 2022, 14(03), 727–735

Publication history: Received on 03 May 2022; revised on 22 June 2022; accepted on 25 June 2022

Article DOI: https://doi.org/10.30574/wjarr.2022.14.3.0541

Abstract

One significant method for developing software is Function as a Service (FaaS), which entails making little, tailored functions to deal with particular jobs. Coders put less emphasis on creating full apps and more on creating these functions that are called when certain events or requests come in. If you want to learn about and use Function as a Service, this article is for you. Customers using the serverless approach do not need to reserve hardware resources, which is a departure from the conventional cloud computing service model. Billing is dependent on real resource consumption, and code execution is event-driven (via HTTP requests, cron jobs, etc.). In exchange, the provider is liable for assigning resources and tasks. Serverless is most often seen as a public cloud service, although there are solutions being worked on and supported by strong players in the industry that will enable private cloud serverless platforms to be built. "Function as a Service" (FaaS), the initial serverless offering, has serious flaws that might cancel out any advantages for providers and customers alike, particularly when it comes to providers' capacity to multiplex resources and customers' ability to save money. Providers and tenants alike could save a ton of money and energy if these problems were solved. In order to prevent serverless from becoming the default cloud computing model, this chapter will provide a thorough overview of its limits and highlight state-of-the-art research to address these issues.

Keywords: Cloud computing; Serverless computing; Security; Threat models; Vulnerabilities

1 Introduction

Cloud computing was once the bee's knees in the information technology industry. Because of this, we were able to separate ourselves from the actual hosting environment. In subsequent iterations, various hardware pieces have been hosted in smaller hosting environments. Virtual computers in the cloud have been operational since the dawn of a new era in the information technology sector with the introduction of virtualization. Subsequently, cloud services ramped up their aggressiveness by introducing PAAS [1]. These are the operating systems that can be run in the cloud. After that, vendors began distributing apps and software through the cloud, relieving customers of concerns about infrastructure and unused resources. The serverless era of cloud computing is here, and no one has noticed. We have now arrived at function-based scaling, sometimes called the serverless architecture [2].

Serverless computing is a relatively new concept that has already sparked heated debate. It provides a worry-free hosting and execution environment, letting developers focus on building and running apps rather than the servers. However, developers are still learning the ropes of this new technology and are still getting a feel for its features and capabilities, particularly in terms of security. So, to help developers understand their security responsibilities in serverless computing and how to maintain the environment safe from attackers, this paper covers these subjects [3].

* Corresponding author: Rajashekar Reddy Yasani

The purpose of this paper is to give readers with up-to-date information about serverless computing by way of a comprehensive survey, so that they may better understand the technology and its advantages and disadvantages. We will continue our examination of the characteristics of serverless architecture, with an emphasis on privacy and security concerns, starting with this architecture [4]. This article compares and contrasts serverless architecture with conventional cloud architecture, delves into the pros and cons of serverless computing, and explains how it influences software development. Additionally, we will introduce serverless enabling technologies and describe their properties. Following this, we will go over some of the current privacy and security concerns with serverless computing, and we will detail any measures taken to address these concerns in order to reduce potential dangers.

1.1 Problem Statement

By eliminating the need to manage servers, serverless computing makes it easier to execute apps. Nevertheless, third parties own the underlying infrastructure, such as servers. Developers lack complete command over the system and data flow, leading to privacy and security concerns [5]. Even while security is always improving and attackers are always changing their tactics, most solutions are still built around old flaws. This means that the issue at hand is distinct from its classical counterpart. Functional as a Service (FaaS) is the foundation of serverless technology, and all of the application-related functions are made public. The fact that it is operating on shared platforms makes it easy for hackers to launch attacks. Static analysis functions, which limit solutions to a small set of policies, are used to create solutions to serverless security attacks. The assault surface is therefore increased. Serverless applications also have minimal monitoring. As a result, keeping tabs on all the attacks is a major worry [6]. Due to its novelty, this technology is particularly vulnerable to zero-day assaults. Subproblems include things like Cross-Execution Data Persistence, Unsecure Third Party Dependencies, and Overprivileged Function Permissions.

2 Background and review of literature

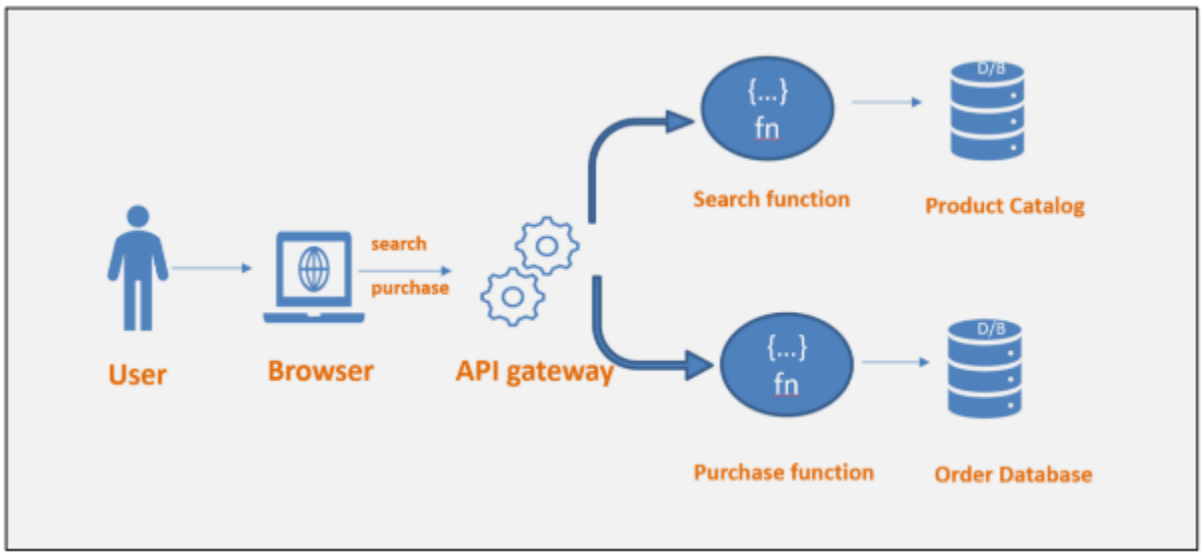

Figure 1 Serverless Architecture

Once upon a time, system administrators would handle everything from allocating RAM and servers to drivers, upgrades, installations, and more. The "Bare-Metal" environment describes this procedure [7]. Afterwards, virtual machines became famous. The ability to easily swap between servers was a huge boon to developers. Technologies for containerization were subsequently employed. This makes it possible for numerous programs to coexist on a single computer without causing any problems. At last, serverless computing emerged in October 2015, thanks to Austen Collins's development and introduction of the concept [8]. The serverless framework is initially built using Node.js using Amazon Web Services Lambda. Use of Function as a Service (FAAS) is essential for this. Inputs are little pieces of code or functions, which are processed, returned, and then shut down. Providers in the cloud handle tasks such as server or cluster deployment, operating system maintenance, patching, capacity provisioning, and server administration for backend components. When adopting serverless, developers don't have to worry about the underlying infrastructure, maintenance, servers, hardware, etc., while they design the solution. Serverless was an inspiration for numerous other providers, who have now developed their own unique approaches and features that can work with a wide range of technologies. This technology is easier to design and manage because to its auto-scaling service. Since serverless is still

in its infancy as a technology, it is not susceptible to many security threats. What follows is a diagram explaining the operation of serverless technology.

Amazon Web Services (AWS) Lambda, Google Cloud Functions, Microsoft Azure, Apache OpenWhist, Tencent SCF, IBM Cloud Functions, Kubeless, Knative, Alibaba Cloud, Spotinst, and Fn are among the most well-known serverless providers [9].

2.1.1 Characteristics of Serverless Computing

Serverless computing is characterized by the following features.

- Stateless: Serverless computing is characterized by its lack of state. Since our own platform generates and deletes code automatically when using Function as a service (FAAS), no data is saved in memory. Therefore, more instances can be registered in this way. The inability to use HTTP sessions is the sole drawback of this feature.
- Efficient: By letting developers only pay for the resources actually used, serverless architectures promote efficiency. The billing approach is based on pay-as-you-go.
- Auto-Scaling: Automatically allocated resources are made accessible as requests are received. One example is Amazon Aurora Serverless, which offers on-demand auto-scaling by responding to application demand for more or less storage space by dynamically starting and stopping databases [10].
- Security: Serverless does its utmost to offer security in every way, which is a major concern these days. By implementing best practices such as frequent patching, utilizing the principle of least privileges, fumigati all inputs, monitoring, and logging services, serverless architecture takes all vulnerabilities into account. Additionally, there is less chance of long-term attacks because the memory is not preserved. Take AWS Lambda as an example; they encrypt all communications using Transport Layer Security [11].
- Debugging: Debugging is made easier for developers using serverless technologies since they can more easily identify mistakes and bottlenecks.
- Language support: Serverless allows developers to work with a wide variety of programming languages. Users have complete freedom in selecting the programming language they choose to use while developing serverless applications. For instance, Go, Ruby, Node.js, Python, Java, and AWS all have support for each of these languages [12]. Plus, they enable runtime API, which opens up more programming languages for them to employ.
- Composability: Easy computation and construction of complicated serverless applications are made possible by the ability to execute functions from other serverless [13].
- Hostless: Users are free from concerns over servers, updates, and security patches with serverless architecture. As a result, the running expenses are cut. With minimal understanding of the application's settings, they can install and execute the application with ease [14].

Functions as a Service (FaaS) is compatible with serverless architectures, as previously stated. The developers or users must first build the function in order to specify the exact goal. Before the cloud service is activated, he specifies the event. The usage of the function instance is checked by the cloud provider. A new function will begin running if it is not used. The application itself displays the outcomes of the conducted functions to the user [15].

3 Faas security: serverless security and mitigation techniques

Security as a service has grown in importance in the cloud computing ecosystem. One form of cloud computing that allows developers to quickly build, test, and deploy application packages is known as Function-as-a-Service (FaaS). Developers may start their development process with just a reliable internet connection and their coding abilities using FaaS, as it removes the stress of managing their own infrastructure. We'll take a break now to walk you through the parts that are coming up. To assist you understand the fundamentals of FaaS (Function-as-a-Service) security, we will answer important questions about it in this article:

The term "FaaS security" is not defined.

In terms of security for FaaS, what are the most important concerns?

● What does FaaS primarily offer?

In what ways can the shortcomings of these services be worsened?

How can these services be secured to the degree that is desired?

Figure 2 Serverless security.

3.1 Enhancing FaaS Security: Serverless Security Best Practices and Risk Mitigation Techniques

Developers have access to strong features with FaaS, an event-driven and immediate execution approach that operates within containers. It makes application deployment easy by removing the need to manage servers. However, these advantages can be diminished if security concerns are disregarded.

3.2 Security Considerations in Serverless Services

The key benefit of serverless computing is that developers may build and maintain apps without worrying about security because the cloud provider bears responsibility for robust security. The following are some of the most prominent security concerns with serverless setups, which are similar to those with on-premises servers:

- Increased Attack Surface

Numerous inputs and event sources, including APIs, IoT devices, and cloud storage, are integrated with serverless computing. Because of the increased attack surface caused by these varied inputs, stricter control and security are required.

- Reduced System Visibility

It can be more difficult to monitor and visualize traffic and inputs/outputs in serverless systems compared to standard on-premise infrastructures. Data breaches might occur due to a lack of visibility. Luckily, these visibility issues have been solved by recent innovations, and they may now be easily integrated into cloud systems.

- Less System Control

Problems may arise while using FaaS products because they involve handing over control of the complete infrastructure to a third party. It may become more difficult to understand and maintain the complete system when using FaaS solutions, which could increase the amount of time spent testing and troubleshooting.

- Essential Knowledge of FaaS Security

A deep understanding of Function as a Service (FaaS) security is essential for realizing FaaS's potential in today's dynamic digital economy. Today, protecting serverless architectures is crucial in a world where technology permeates every part of our lives. With this essential security information in hand, enterprises, developers, and system administrators can strengthen applications, safeguard sensitive data, and keep up with the ever-changing regulatory compliance landscape. Since they both deal with comparable issues, serverless security needs are quite similar to SaaS security measures.

Additional resources for learning about cloud computing services include:

- PaaS security
- IaaS security

Understanding how to maintain high levels of security is vital when dealing with serverless systems and FaaS services. This necessitates learning about the most significant security holes and filling them with proven solutions.

4 Common faas security vulnerabilities

4.1 Vendor Lock-in

Issues with vendor lock-in, in which users are unable to transfer products or services because of inherent or technical hurdles, are common in FaaS security. This reliance can be especially troublesome in the FaaS environment. You can design and deploy functions that are more scalable and adaptable across diverse contexts by employing open-source products, which reduces the danger of vendor lock-in and mitigates this worry.

4.2 Insecure Serverless Deployment Configurations

Hackers can easily gain access to serverless applications with insecure configurations, making them much more vulnerable. Denial of service attacks and problems related to incorrectly specified timeout settings are common types of threats.

4.3 Broken Authentication

Serverless apps are vulnerable to attacks like credential theft or broken authentication. A compromised authentication could allow hackers to access one function and potentially compromise the entire system, since serverless apps depend on microservices.

4.4 Sensitive Data Exposure

Any information that requires extra precautions to prevent disclosure or misuse is considered sensitive data. Information ranging from login credentials to personally identifiable information is considered sensitive in the FaaS environment. Unsecure databases and improperly setup cloud storage are two examples of the kinds of problems that can cause vulnerabilities that could lead to disclosure.

4.5 XML External Entities

A malicious actor can exploit XML external entities (XXE) attacks to influence how an application processes XML data. Intruders can gain access to the application server's file system and, in turn, any backend systems that the application uses.

4.6 Broken Access Control

While administrators usually keep an eye on access control, problems with this feature can arise and quickly become significant threats to the security of your software as a service (SaaS). Some users are able to access resources and execute unauthorized actions due to these flaws, which show up as broken access control vulnerabilities.

4.7 Cross-Site Scripting

A major threat to FaaS security is cross-site scripting (XSS), which necessitates strong countermeasures. When a hacker attaches malicious code to a seemingly genuine website, it can infect a victim's browser and cause cross-site scripting (XSS).

4.8 Insecure Deserialization

When an attacker or website analyzes data that the user controls, this is known as insecure deserialization. This allows the attacker to possibly execute damaging functions and code by manipulating the serialized data objects. When untrusted data is used to exploit the application's logic, it is considered insecure deserialization in the context of FaaS.

4.9 Using Components with Known Vulnerabilities

When developers employ unsafe components or accidentally produce susceptible code fragments, they are usually the ones to blame. Depending on the affected component, these vulnerabilities might cause a wide range of issues. One way to lessen the impact of this is to only get software development components from trusted sources.

4.10 Unauthorized Access and Data Exposure

When an unauthorized individual acquires access to a company's data, this is known as unauthorized access. Data copying and theft from the database is one of the most common things that attackers do when they exploit this access.

4.11 Injection Attacks and Code Dependencies

Any system or function, including those in the FaaS ecosystem, could be compromised through a dependency vulnerability. Additional dependencies inserted into the application can lead to code dependency vulnerabilities, which can severely compromise FaaS security.

5 Best practices for faas security

5.1 Security Authentication and Authorization Mechanisms

You can confirm a user's identity and find out what they can do in an app with a safe authentication method. Even though most people employ a simple username and password configuration, it has a lot of restrictions. These days, cutting-edge security measures like SSO, Two-Factor Authentication (2FA), and Multi-Factor Authentication (MFA) are used by online platforms to ensure that they are completely secure.

5.2 Data Security in Function-based Apps

Data security includes a wide range of procedures in function-based applications. Data encryption utilizing recognized technologies is an important part of a dependable approach. This makes sure that data is stored or delivered in a secure format. Threat actors will still be unable to use the data for their malicious purposes if they manage to breach the application or platform.

5.3 Security Insights and Log Monitoring

Security for FaaS relies heavily on log monitoring and analytics, which can be made easier with intuitive tools. Insights into application security are generally pre-installed on modern cloud platforms, making them simple to implement and use. Log monitoring entails gathering data from many sources, analyzing it, and then acting on the results.

5.4 Implementing Proper Identity and Access Management (IAM)

Almost every company is now implementing some kind of sophisticated Identity and Access Management (IAM) strategy. Implementing robust IAM systems to manage and protect all elements is essential in today's digitally connected environment, where numerous cyber risks are present. This is to ensure consistency and security. Most identity and access management (IAM) solutions come as a bundle that covers all the bases, meeting all the reliable FaaS security needs all in one place.

5.5 Encrypting Data in Transit and at Rest, and Using Secure Key Manager

Protecting data at rest and while in transit are two sides of the same coin when it comes to data security. Data encryption while in transit is an essential component of all cloud services. The usage of secure key management solutions allows developers to improve and maintain the security of data transport.

5.6 Mitigation Techniques for FaaS Vulnerabilities

Service providers and end users alike are at risk from security flaws in FaaS platforms because of the nature of these systems. An all-encompassing plan is necessary to successfully counter these weaknesses. Here we will give you some powerful and useful tips to get the FaaS security you want:

5.7 Serverless Security Tools and Services

In the same way that on-premise infrastructures provide a variety of security tools and services, serverless platforms do the same for enterprises of all sizes. Log analytics and serverless monitoring are all part of these products. To further evaluate and keep tabs on your apps' security, you can incorporate container security checkers into your projects if you so desire.

6 Isolation and containerization

One of the most important aspects of container technology's security is isolation. Linux containers use a number of ways to provide isolation, and you can adjust the degree of isolation to suit your needs. To further protect containers from harmful functionalities, some isolation techniques are used. Modern methods have the potential to improve container isolation without sacrificing efficiency.

6.1 Input Validation and Sanitization

To make sure user inputs are up to par, input validation is an essential process. As part of the sanitization process, these inputs are checked for and any undesirable or hazardous characters are removed. Among the many problems that might result from these undesired characters is the ability for bad actors to use website input areas to execute SQL queries and scripts.

6.2 Stateless Key Management

To secure keys, it is just as important to use strong cryptographic methods. You must not discount the security of highly dependable keys just because they are very difficult to crack. It only takes minutes for hackers to obtain an encryption key, which means they can possibly destroy your entire organization by getting to your functionalities.

6.3 Data Protected at Point of Ingestion

Data ingestion and transit security measures are multi-faceted. Using secure protocols and channels is one viable method. Authentication, authorization, integrity, and secrecy are core safety elements of secure protocols. Several protocols and channels for secure communication are now at your disposal, including HTTPS, SSL/TLS, SSH, and SFTP.

6.4 Compliance and Regulatory Considerations

Organizations must conform to regulations, norms, and laws to avoid fines and other negative outcomes; noncompliance can have the same effect. Concerns about cybersecurity can be alleviated and worries about FaaS security can be put to rest with a trustworthy regulatory framework.

7 Faas security in the context of data privacy regulations

Ensuring the security of FaaS requires knowledge of and experience with industry-specific regulatory requirements. The ever-changing threat landscape, defined by advanced assaults on FaaS products, has increased the importance of compliance in FaaS security.

7.1 Auditing and Compliance Monitoring

An essential aspect of this process is having cybersecurity service providers perform third-party audits. These audits will help find and disclose any concerns that are related to cybersecurity. Your company will also continuously follow these regulatory frameworks with the use of software that monitors compliance.

7.2 Trending Topics in FaaS Security

Even while serverless computing is making developers' lives easier, there is still a lot of interest in finding ways to make servers run quicker. Recent web searches have shown a considerable amount of interest in serverless computing's integrated AI solutions and real-time monitoring tools.

Innovations in FaaS and DevOps Security

To make the most of FaaS adoption and create a safe environment, it is important to implement security improvements.

Here are a few of the most well-known developments in FaaS security:

- Monitoring and analyzing logs without servers
- Environments with hybrid and multi-cloud capabilities
- Practices involving zero-trust

7.3 Anticipating Emerging Threats

Although it may be time-consuming and labor-intensive, planning for potential dangers in the future pays off in the end. Predicting these new cybersecurity risks should be a top priority for any successful strategy, since they have the ability to stunt an organization's progress.

8 Conclusion

Function-based services in the cloud need special protection, and that's exactly what FaaS and DevOps security are trying to give. Users and organizations can still benefit greatly from FaaS systems, despite the fact that they increase the attack surface and decrease system visibility. For optimal performance, a hybrid model might be the way to go. This research has demonstrated both the benefits and the drawbacks of serverless computing. When it comes to protecting their work, users and developers that utilize serverless technologies bear a disproportionate amount of responsibility. Users and developers alike need to take extra precautions to safeguard their environments, particularly against input-oriented attacks and maintenance gaps that could let in harmful components.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Adam, J. (2019, March 28). The serverless evolution of big data & AI apps. K&C Consulting. https://kruschecompany.com/serverless-big-data-ai/

[2] Ahmed, A. (2017, March 6). Have an account with Uber or FitBit? You better change your password TODAY! Retrieved March 3, 2021, from https://www.linkedin.com/pulse/have-account-uber-fitbit-you-better-change-your-password-abdi-ahmed

[3] Allen, L. (2022). AWS Lambda–serverless architecture done right. Retrieved January 17, 2022, from https://www.missioncloud.com/blog/aws-lambda-serverless-architecture-done-right

[4] Alpernas, K., Flanagan, C., Fouladi, S., Ryzhyk, L., Sagiv, M., Schmitz, T., & Winstein, K. (2018). Secure serverless computing using dynamic information flow control. *ArXiv:1802.08984 [Cs].* Retrieved from http://arxiv.org/abs/1802.08984

[5] Amazon Aurora Serverless. (2021). MySQL postgreSQL relational database. Amazon Web Services, Inc. Retrieved February 23, 2021, from https://aws.amazon.com/rds/aurora/serverless/

[6] Amazon EventBridge. (2021). Event bus. Amazon Web Services, Inc. Retrieved October 17, 2021, from https://aws.amazon.com/eventbridge/

[7] Amazon Kinesis Data Streams—Data streaming service. (2021). Amazon Web Services, Inc. Retrieved October 17, 2021, from https://aws.amazon.com/kinesis/data-streams/

[8] Amazon SQS. Message queuing service. (2018). Amazon Web Services, Inc. Retrieved October 17, 2021, from https://aws.amazon.com/sqs/

[9] Amazon Web Services (AWS)—Cloud computing services. (2021). Amazon Web Services, Inc. Retrieved April 1, 2021, from https://aws.amazon.com/

[10] Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., … & Suter, P. (2017). Serverless computing: Current trends and open problems. ArXiv:1706.03178 [Cs]. http://arxiv.org/abs/1706.03178

[11] Bertram, A. (March 22, 2021). Choose the right Google Cloud serverless service. Search Cloud Computing. Retrieved October 1, 2021, from https://searchcloudcomputing.techtarget. com/answer/Choose-the-right-Google-Cloud-serverless-service

[12] Beswick, J. (2020, July 29). Building a serverless tokenization solution to mask sensitive data. Amazon Web Services. https://aws.amazon.com/blogs/compute/building-a-serverlesstokenization-solution-to-mask-sensitive-data/

[13] Capital One reports inside job data breach. (2017, August 3). ITRC. Retrieved January 23, 2022, from https://www.idtheftcenter.org/post/capital-one-reports-inside-job-data-breach/

[14] Catalin C. (2019). Over 100,000 github repos have leaked API or cryptographic keys. ZDNet. Retrieved June 5, 2022, from https://www.zdnet.com/article/over-100000-github-reposhave-leaked-api-or-cryptographic-keys/

[15] Chris. (2021). # serverless-from the beginning, using Azure Functions (azure portal), part I. Retrieved June 16, 2022, from https://softchris.github.io/pages/serverlessone.html#serverless-on-azure