Security threats in serverless

Daniel Gustafsson & Abdelmoujib Megzari

May 2022

1 Introduction

Before the Cloud Era, companies used to host their services on-premise on computers and servers that the company owned and maintained. The setup and maintenance of these computers and servers were costly and meant that the companies had to do extra work other than coding. Therefore cloud providers came up with solutions for these companies to outsource this maintenance. Serverless comes as a Function As A Service solution (Faas) figure 1 which came historically after Infrastructure As A Service (Iaas) and Platform As A Service (Paas). Today serverless architecture is a broadly used solution. Its usage has completely exploded since the launch of AWS Lambda in 2014 [10]. According to a survey by O'Reilly in 2019 [6], as many as 40% of companies have adopted serverless in their organization, and almost all of them consider it a successful adoption.

Serverless architecture is considered a secure solution compared to its predecessors due to many possible security aspects managed by serverless providers. This makes many people neglect the security aspect when using such a solution. However, the solution isn't completely secure since the serverless providers can not counter many threats that only the company can anticipate and protect itself from.

In this essay, we aim to raise awareness about such security aspects by presenting some of the most common security Threats from a serverless security perspective. We also try to present a non-exhaustive list of prevention technics for each security vulnerability. We start first by giving the reader some background information to the reader by briefly introducing the serverless Architecture. We then go on to present three different threats, which are: broken authentication, Denial of service attacks and financial resources Exhaustion, and finally, the Over-Privileged functions Threat. for each of these problems, we start by presenting the threat and then proposing some solutions and prevention techniques.

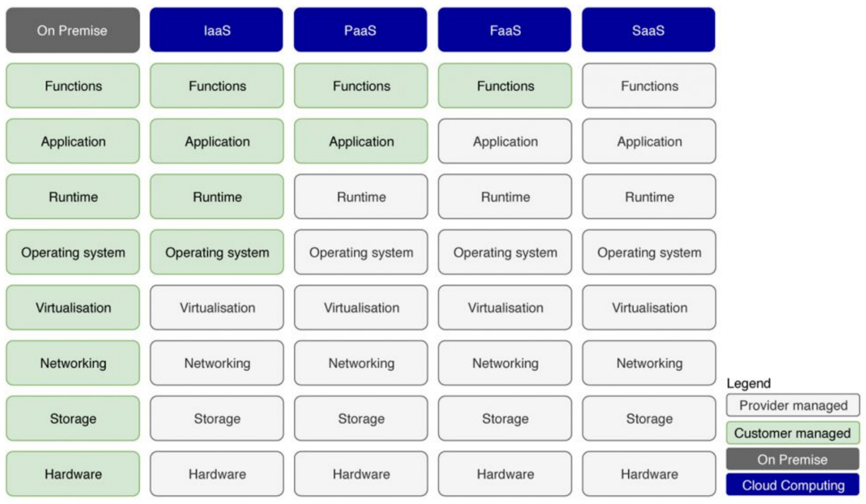

Figure 1: Cloud services

2 Background

The concept of serverless is to not run a traditional server which handles HTTP requests and database access etc. Serverless is instead like a time-share of servers, you define some functions2 and when those functions should run and how much resources you want to assign to those functions and then the cloud provider solves the rest.You can define multiple types of triggers or your functions, such as an HTTP request, a time trigger, a change in a file system, or a call from another function. Serverless architecture also has the advantage of autoscaling. When your service has lower traffic, the servers in the cloud providers data center are probably providing services to some other organization and when traffic on your services are increasing, more resources will be allocated to you again.

All the big cloud providers run a payment model where you pay for what you use, this lowers the threshold for smaller companies significantly. A company no longer needs to pay for expensive servers and other hardware to offer a web service. The company instead just pays for the traffic it gets.

Similarly, a company running a serverless service can easily change the resource allocation of each microservice which makes it easy to limit the cost from the cloud provider [1]. This can of course later be changed to scale up the service, making scalability another huge perk of serverless.

Serverless also has security benefits over other server-based solutions [7] [4]. All of the big cloud providers use encrypted traffic and storage as well as supplying robust authentication systems. Before serverless, it was up to the companies to solve these problems themselves which often led to unencrypted databases, passwords saved in plaintext [8] etc.

To sum it up, serverless removes many headaches from the user and reduces cost of entry to deploy a web application. But serverless is not perfect, there are security threats associated with using it which needs careful consideration.

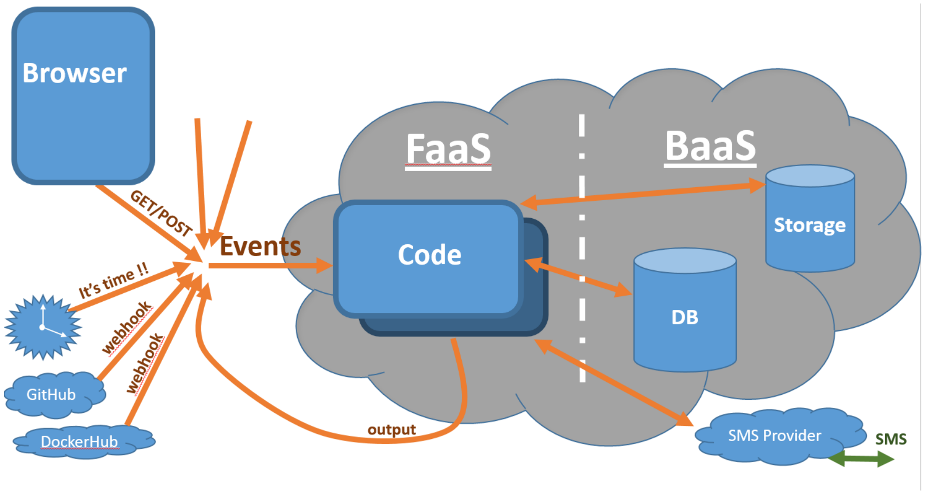

Figure 2: Serverless architecture

3 Security threats in serverless

In this section we will present some of the major security threats associated with serverless. The threats are Broken Authentication, DOS & Financial Resource Exhaustion and Over-Privileged Functions and will be presented in that order.

3.1 Broken Authentication

3.1.1 Problem

Serverless application are stateless and are often architected with a microservice-like system design. This means that each independent function is exposed to the world and is a potential gateway for attackers. If just one of these function's authentication is mishandled it will impact the whole application. That mishandled function could be found by attackers and then used to gain access to data which they are not authorized for.

It is not only external-facing resources that can be exploited, if for example a function is triggered by organizational emails, an attacker might send spoofed emails to trigger the function and execute internal functionality without authentication [9].

3.1.2 Solution

To protect your application from these kind of intruders, make sure to always be strict and never compromise in function authentication. No matter how small the function is, the authentication scheme with proper access control is crucial.

You should always try to follow the best practice of your chosen cloud provider, all of the big ones have their own authentication solutions which could prevent broken authentication attacks [9]. For external facing resources, authentication should always be required and internal resources should follow best practices like encrypted channels, password and key management or client certificate.

Even though it may take up some time, it might be worth going through authentication schemes from time to time and check so that everything is in order. This is something that managers at companies have complained about not having the time and resources to do but many can agree that having your product be destroyed by hackers or you sensitive data leaked to the public is worth putting time and resources avoiding.

3.2 DOS & Financial Resource Exhaustion

3.2.1 Problem

A denial of service attack is an attack that targets making a service unavailable. It's usually done by flooding the server with requests or by flooding the network making the service unavailable for a certain period. While these attacks aren't explicitly related to serverless. Their effect on serverless architecture can be different. Since the payment is made based on the number of executions, even a nonsuccessful Denial of service attack can financially harm the serverless user. However, the autoscaling aspect of a serverless deployed architecture can make it harder to deny service. It can hit your wallet hard due to this same aspect. The term Denial of wallet can be used in this case[5][11]. It's still possible to make a service totally unavailable because of the limited number of instances that can be run in a specific serverless plan.

Many different types of denial on a serverless architecture can exist.

For example, an attacker can exploit a wrong configuration or a vulnerability in the function code to cause a function to trigger itself endlessly, thus creating an endless loop.

Another attack can be a one-hit knockout which is finding that single function call that will cause the function to use a lot of resources.

There is also the distributed denial of service attack in which a hacker manages to compromise many computers, thus creating what is called a botnet. He can then use this botnet by ordering all computers to fetch the service at the same time. This type of attack is even harder to detect because it's impossible to distinguish the legitimate request from the illegitimate ones

3.2.2 Causes and solutions [2]

Many factors can make a serverless system vulnerable to a denial of service attack. The most important one is access permissions management. It's crucial to be aware of who can execute your function. A good practice is to give a minimum of permissions by default and add permissions only when needed.
It's also essential to keep your credentials and API keys secret. A good practice is using different keys for different services and employees and changing them regularly.
An interesting way to counter distributed denial of service can be using a leverage API Gateway to manage the serverless function's access. Users can use it to authenticate before using your function, and the gateway can decide whether or not to allow this access. The gateway itself can be a victim of a denial of service attack, but it can limit the impact by being used before functions that will use more computations. and there are also some known solutions to protect the gateway from these attacks that aren't related to serverless.
Another good solution is to use security software designed specifically to prevent a DDoS attack on serverless architecture.But a good practice is not to rely only on such softwares
Finally, it's good to set limits for resource usage and function timeout to prevent such attacks. Setting Billings alerts can also help you detect abnormal service usage and act accordingly.

3.3 Over-Privileged Functions

3.3.1 Problem

Similar to broken authentication, over-privileged functions can be a critical security threat in a serverless application. If a function has privilege to access more data than it should, that function could be abused to leak or modify that data.

Consider a scenario where there exists a undiscovered vulnerability in a package which allows code injection. The function's purpose is to get a list of items from a specific table connected to a user in the database. The developers have been lazy and just assigned the function to have all database privileges (common occurrence). A potential attacker can now both read and write the whole database since the function is over-privileged, having potentially catastrophic consequences.

This is the kind of scenario you want to avoid in your application. It might be hard to control the security from imported packages but you can control the privilege your serverless functions has.

Another problem with this is that many applications use hundreds of independent functions and it can seem overwhelming to go through them all to check the privileges [3]. There is unfortunately no easy solution to this.

3.3.2 Solution

The principle of "Least Privilege" should always be followed. If a function is used to get some data from the database, it should only have access to the specific table which it gets from and should only have read access. This is true for all functions and a function should never have a privilege which it won't use. This is an easy way to keep damage at a minimal when your service is being attacked.

If the developers from the scenario above would have followed these principles, the potential attackers would have instead just been able to read a specific table in the database. This is obviously a bad outcome as well but magnitudes better than in the over-privileged scenario. Some companies are able to get through a cyber attack without much damage to their brand and others are completely ruined, these kinds of precautions might make that difference.

4 Conclusion

It is clear that serverless has some issues but since most of these issues come from wrongful configuration, these issues can generally be avoided by being extra careful when configuration your serverless service. With all the security benefits that come with serverless like standardized and up to date authentication, encrypted storage, automatic application of security patches and more, the security threats that are added by using serverless seems quite minimal.

With all this in consideration it is our conclusion that serverless is most likely a positive thing for your organizations cyber security. It should be easier to configure serverless to avoid broken authentication, denial of service attacks or over-privileged functions then to not use serverless and instead have to implement all the now missing security features of serverless yourself.

With the additional perks of reduced cost, faster and easier deployment, automated scalability and to avoid the headaches associated with owning hardware, using serverless should be an easy choice for pretty much anyone.

4.1 Reflection

The topic of serverless is very modern and relevant and it has been a very interesting learning experience to read about and gain insight on the security aspects of it. We thought it was especially interesting that most of the security threats in serverless are derived from misconfiguration and not from flaws in serverless itself. Maybe the true flaw in serverless is that it is so easy to misconfigure it? A well designed system should probably be easy to use, otherwise people probably wont use it correctly. That is a question for another essay.

References

[1] Ioana Baldini et al. “Serverless Computing: Current Trends and Open Problems”. In: *Research Advances in Cloud Computing*. Ed. by Sanjay Chaudhary, Gaurav Somani, and Rajkumar Buyya. Singapore: Springer Singapore, 2017, pp. 1–20. ISBN: 978-981-10-5026-8. DOI: 10.1007/978-981-10-5026-8_1. URL: https://doi.org/10.1007/978-981-10-5026-8_1.

[2] Aaron Chichioco. “Going Serverless? Common Serverless Security Issues and Best Practices by Aaron Chichioco”. In: (2019). Accessed: 2022-05-03. URL: https://hakin9.org/going-serverless-common-serverless-security-issues-and-best-practices/.

[3] Sam Flaster. “Why Tackling Serverless IAM Threats Takes a Team”. In: (2021). Accessed: 2022-04-29. URL: https://www.cyberark.com/resources/blog/why-tackling-serverless-iam-threats-takes-a-team.

[4] Hassan B. Hassan, Saman A. Barakat, and Qusay I. Sarhan. “Survey on serverless computing”. In: *Journal of Cloud Computing* 10.1 (2021). DOI: 10.1186/s13677-021-00253-7.

[5] Daniel Kelly, Frank G. Glavin, and Enda Barrett. “Denial of wallet—Defining a looming threat to serverless computing”. In: *Journal of Information Security and Applications* 60 (2021), p. 102843. ISSN: 2214-2126. DOI: https://doi.org/10.1016/j.jisa.2021.102843. URL: https://www.sciencedirect.com/science/article/pii/S221421262100079X.

[6] Roger Magoulas. “O´Reilly serverless survey 2019: Concerns, what works, and what to expect”. In: (2019). Accessed: 2022-05-02. URL: https://www.oreilly.com/radar/oreilly-serverless-survey-2019-concerns-what-works-and-what-to-expect/.

[7] Vitaly Makhov. “Server vs. Serverless: Benefits and Downsides”. In: (2021). Accessed: 2022-05-02. URL: https://nordicapis.com/server-vs-serverless-benefits-and-downsides/.

[8] Alena Naiakshina et al. “Why Do Developers Get Password Storage Wrong? A Qualitative Usability Study”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’17. Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 311–328. ISBN: 9781450349468. DOI: 10.1145/3133956.3134082. URL: https://doi.org/10.1145/3133956.3134082.

[9] OWASP. “OWASP Top 10 Interpretation for Serverless”. In: (2017). Accessed: 2022-05-03. URL: https://owasp.org/www-pdf-archive/OWASP-Top-10-Serverless-Interpretation-en.pdf.

[10] Archit Rathi. “The rise and rise of serverless”. In: (2021). Accessed: 2022-05-02. URL: https://www.oxx.vc/post/the-rise-and-rise-of-serverless.

[11] Summit Route. "Denial of Wallet Attacks on AWS". In: (2020). Accessed: 2022-05-03. URL: `https://summitroute.com/blog/2020/06/08/denial_of_wallet_attacks_on_aws/`.