Review of Serverless Computing Database Security

Karis Kim

Kennesaw State University

IT6863 Database Security and Auditing

Dr. Nick Suppiah

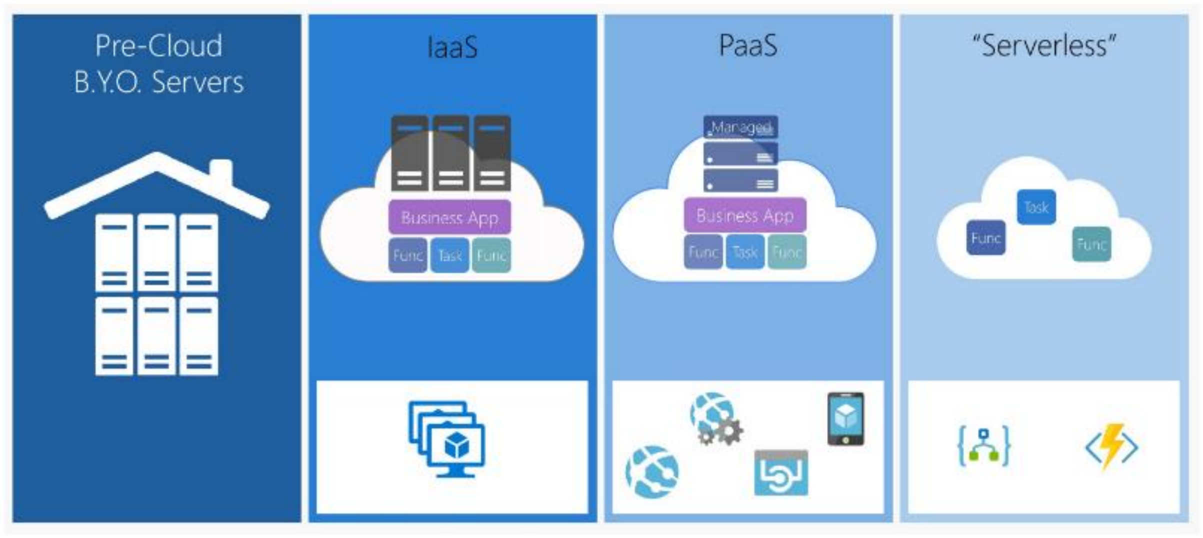
December 5, 2018

Table of Contents

Review of Serverless Computing Database Security

Serverless Computing Introduction

The age of cloud computing has recently led the way to build and run applications without the hassle of managing servers, provisioning, or scaling, known as serverless computing ("Serverless Computing", n.d.). In serverless computing, applications are hosted by some cloud provider like AWS, who will manage all the server software and hardware, so that the client can focus on front end application functions that are invoked individually by function (Prothero, n.d.). This is also known as Faas (Function-as-a-Service), because individual functions of an application are separately deployed by the cloud provider. Serverless computing or FaaS is also known as event-driven computing, because individual stateless functions are uploaded by events or triggers and executed when those events occur, liberating the developer from the need to bother with server matters like load balancing and scaling (Simi, 2018; Varghese & Buyya, 2018). Unlike traditional cloud computing, serverless computing is billed on executions by function rather than by server instances that your entire application occupies, irrespective of actual execution duration (Watson, 2017). Serverless architectures are now offered by many cloud providers, such as AWS Lambda, Google Cloud Functions, IBM OpenWhisk Cloud Functions, Microsoft Azure Functions, and even Oracle with Fn Project. Since its debut around 2014 with AWS Lambda (Roberts, 2018), serverless computing is quickly gaining popularity as the latest cloud innovation, because it enables businesses to push applications into production much quicker and more easily, since developers do not need to spend time on server issues (Savage, 2018).



Reprinted from Watson, M. (2017, May 15). What Is Function-as-a-Service? Serverless Architectures Are Here! Retrieved November 8, 2018, from https://stackify.com/function-as-a-service-serverless-architecture/

Scope and Objectives

This paper is an overview of what serverless computing entails, its benefits and drawbacks, how serverless architecture allows applications and services to be provided, what the state of database security is in serverless computing, and what options are currently available for serverless database security. Serverless in this paper refers to the latest innovations of Function-as-a-Service in event-driven cloud computing, rather than serverless in the context of RFID, though there may be more peer-reviewed scholarly articles available for serverless for RFID. While FaaS overlaps in part with PaaS (Platform-as-a-Service) or Baas (Backend-as-a-Service), the scope of this paper will be limited to the current general reference to FaaS regarding serverless applications.

Benefits of Serverless Computing

In determining the benefits of serverless computing, the areas of consideration are: application design and application deployment and operations. In terms of design, serverless computing lightens the developer's workload because the server application logic, such as configuring compute resources for load balancing and handling event queues for scaling, is borne by the cloud provider (Google Cloud, 2017). Without the need for developers to expend their resources on such backend server coding, businesses can launch applications to the market much more rapidly. This reduced development time not only allows rapid push to market, but also offers development cost savings to the businesses.

In terms of deployment and operations, one of the most, if not *the* most, prominent benefits of serverless computing is that scaling is automatically and elastically managed by the cloud provider. Once your serverless application is deployed, particularly if your application has unpredictable traffic requirements, developers and businesses can be completely free from the worries and hassles of server provisioning. According to Roberts (2018), "a fully serverless solution requires zero system administration" and that is a major bonus for deployment and operations. The cloud provider assumes "the responsibility for datacenter management, server management and the runtime environment" (Lynn et al., 2017). Zero system administration is not only a benefit in time, mental health of developers, and management cost, but also a savings in actual operation cost, because serverless models charge only for the function or events that are used, rather than entire server instances that may be idle at times.

Moreover, serverless can be very useful for performing regularly scheduled or event-driven analysis on datasets, cleaning up log or IoT data before processing or transmitting, and automating backups or daily processing tasks (Rogers, 2017). Dwyer (2016) lists the following examples as suitable candidates that can benefit from serverless: backend transactions, batch processes, data pipelines, bots and scheduled jobs.

Drawbacks of Serverless Computing

As stellar as the benefits of serverless computing appear to be, the drawbacks of serverless computing are quite dreary. Since the bottom line sustaining the popularity of most industry innovations is cost efficiency (i.e. popular innovations will not be able to maintain rapid growth in popularity if the cost is too high for businesses to bear), the first drawback of serverless listed here shall be the cost. Ironically, one of the main promotional attractions of serverless architecture is the cost savings of pay-per-function pricing model over traditional pay-per-instance models. However, a comparative research of serverless computing environments by Lee, Satyam, and Fox (2018) revealed that the prevalent promotional claims of lower costs with pay-per-execution models in serverless computing may not be the case for all types of applications. Lee et al. found that in a cost comparison between serverless computing and traditional virtual machines, the cost in seconds for serverless is almost 10 times more expensive, with regard to allocated compute resources for sequential functions (2018). As with Lee et al.'s findings, Eivy (2017) cautions the potential "economics of serverless cloud computing," and reveals that the relative cost of serverless depends on the scale of the application, where the larger the application and HPS (Hits Per Second), the more expensive serverless costs will be in comparison to traditional virtual machines. Furthermore, in serverless pricing, as with other cloud service pricing, users must be cognizant of variable billable units, additional charges and frequent pricing changes.

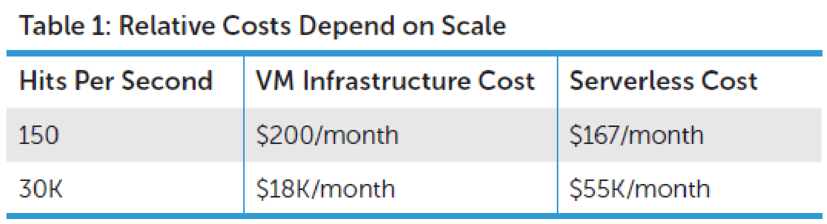


Table 1: Relative Costs Depend on Scale

Hits Per Second	VM Infrastructure Cost	Serverless Cost
150	$200/month	$167/month
30K	$18K/month	$55K/month

Reprinted from Eivy, A. (2017). Be Wary of the Economics of "Serverless" Cloud Computing, IEEE Cloud Computing, *4(2), 11. http://doi.ieeecomputersociety.org/10.1109/MCC.2017.32*

In addition to pricing concerns, the following drawbacks inherent to serverless architecture are expounded by Roberts (2018). In serverless, the server administration is relinquished to the cloud provider, so a third-party vendor has control of your system and you do not (Akiwatkar, 2017; Roberts, 2018; Simi, 2018; Thomas, 2018). Loss of system control could result in "system downtime, forced API upgrades, loss of functionality, unexpected limits, and cost changes" (Akiwatkar, 2017), since vendors will be prone to make decisions that allow them to deliver realistic goals and balance the interests of multiple customers over one individual customer (according to Charity Majors as cited in Roberts, 2018).

As with traditional cloud services, serverless also poses multi-tenancy risks with security and performance (Akiwatkar, 2017; Roberts, 2018), which will be further discussed in the next section of this paper. Another drawback is that serverless functions "will be implemented differently by another vendor" (Roberts, 2018), making it very difficult, if not impossible to change to serverless vendors without changes to the design or architecture (Simi, 2018). This is often referred as vendor lock-in (Akiwatkar, 2017; Roberts, 2018; Simi, 2018). In serverless, coding and coordination will need to be replicated across each of the clients if the database needs to be migrated, whereas in traditional architecture, this repetition would not have been necessary (Roberts, 2018).

Not only does serverless create vendor lock-in, but also, the developer does not have the option “to optimize [the] server design for client performance” (Roberts, 2018). Developers must also depend on the serverless vendor for debugging and monitoring of distributed serverless systems (Akiwatkar, 2017; Roberts, 2018). Serverless vendors like AWS Lambda place a limit of 5 minutes for a function execution (Eivy, 2017), so another drawback is that serverless is not ideal for long term tasks like uploading videos. Akiwatkar (2017) lists implementation drawbacks like difficulty with integration testing and versioning as well.

Database Security Issues in Serverless

Contrary to the immediate implication of the name “serverless” computing, serverless applications do not exist independently of servers; servers still remain a necessary part of serverless applications. In serverless, though, the serverless vendor rather than the application developer/owner is tasked with “securing the data center, network, servers, operating systems and their configurations” (Segal, 2018). The entire premise of serverless computing is that the developer/owner of the application does not need to be concerned with any matters of the server, since that responsibility is delegated to the serverless provider. So under the lull of that convenience, it is not hard for businesses to fall under the misguided impression that security is provided by implementing “a bunch of ‘best practices’ into your organization” which may be “considered outdated or even actively dangerous” (Topper, 2018) and/or that security measures placed by cloud providers to meet minimum compliance requirements will be sufficient as security controls for my serverless application. However, Patrizio’s article reports that the security firm PureSec performed an audit of over 1,000 serverless applications and found that “one in five serverless apps has one form or another of a critical security flaw, allowing attackers to manipulate applications and perform various malicious actions” (2018). In a PurSec whitepaper, Segal (2018) presents the best summary of the following database security drawbacks in serverless architectures:

- “Larger attack surface” for potential security breaches (Roberts, 2018; Segal, 2018)
- Attack surface is very complex in serverless (Segal, 2018)
- System complexity makes monitoring difficult (Segal, 2018)
- Security testing in serverless is more complex
- Auto-scanning tools like DAST, SAST, IAST cannot be used (Segal, 2018)
- Traditional security layers like firewalls cannot be used (Segal, 2018)

The top ten of the most critical security risks in serverless architectures in order of severity according to Segal (2018) are:

1. Function Event Data Injection
2. Broken Authentication
3. Insecure Serverless Deployment Configuration
4. Over-Privileged Function Permissions & Roles
5. Inadequate Function Monitoring and Logging
6. Insecure 3rd Party Dependencies
7. Insecure Application Secrets Storage
8. Denial of Service & Financial Resource Exhaustion
9. Serverless Function Execution Flow Manipulation
10. Improper Exception Handling and Verbose Error Messages

Serverless applications are basically event-driven functions, and many types of event sources can trigger the function of the serverless application, which is why the attack surface is increased and becomes complex (Segal, 2018). Some of the most common examples of function event data injection flaws listed by Segal (2018) include: OS command injection, function runtime code injection, SQL/NoSQL injection, Pub/Sub message data tampering, object deserialization attacks, XXE, and SSRF.

In addition to injection attack vulnerabilities, thoroughly applying a strong authentication system in serverless applications is extremely complex and the authentication systems can easily be remiss. This is an inherent security vulnerability because serverless applications are a conglomeration of "oftentimes contain[ing] dozens or even hundreds of distinct serverless functions, each with its own specific purpose" (Segal, 2018), so orchestrating proper access protections to all of those functions, events or triggers is bound to leave vulnerabilities open. With an inadequate authentication protection, a malicious attacker may be able to access and manipulate the application logic and corrupt the application. A related potential security risk with authentication is granting over-privileged permissions via a single permission model, as a result of trying to escape the complexity of having to manage individual function permissions to hundreds of serverless functions (Segal, 2018). That would violate the basic security concept, the principle of least privilege, and potentially endanger the security of the entire system.

Another major security vulnerability in serverless computing arises from the fact that serverless functions are stateless and "rely on cloud storage infrastructure to store and persist data between executions" (Segal, 2018). Coupled with the multi-tenancy problem mentioned in the previous section discussing drawbacks of serverless, it is not surprising that insecure serverless deployment configurations have resulted in "numerous incidents… which ended up exposing sensitive confidential corporate information to unauthorized users" (Segal, 2018). Compared to traditional infrastructures, serverless applications require each serverless service or function to have “its own secure deployment configuration” (Segal, 2018), and since serverless is so new, there is a lack of general standards regarding how to accomplish a secure serverless configuration for deployment.

Moreover, any security breach attempts that could be detected and prevented with good real-time security monitoring and logging are not possible in serverless applications that reside in distributed serverless cloud centers rather than on prem (Segal, 2018). While some vendors provide a certain amount of serverless monitoring, serverless application developers/owners still remain dependent and limited by the vendors (Roberts, 2018).

Current Solutions for Database Security in Serverless

While the industry may generally agree that serverless computing will accelerate down the path of adoption, particularly “given that billions of devices will need to be connected to the edge of the network and data centers” (as cited by Varghese & Buyya, 2018), it is currently difficult to even locate solutions available for addressing database security in serverless architecture. Serverless computing has only been around for less than 5 years, with major cloud competitors like Google, IBM and Microsoft joining the market only about 2 years ago after Amazon Lambda’s lead (Lynn et al., 2017); but, that does not excuse or mitigate the fact that an ever-growing need for database security solutions for serverless has yet to be addressed. As the leader and first provider to launch serverless computing, Amazon Web Services provides a couple of serverless security measures. PureSec, a serverless security firm based in Tel Aviv, Israel and a partner network with AWS, offers SSRE (Serverless Security Runtime Environment) and SSP (Serverless Security Platform) (PureSec, 2018).

AWS IAM (Identity and Access Management) API is provided to allow users to perform audits periodically to determine if some policies need to be changed with updated access information and/or if some policies have over-privileged permissions (Cisco, 2017). The Cisco whitepaper (2017) states that AWS IAM APIs and Amazon Lambda can work together to show what permissions a particular function has granted, thereby limit API calls and their scope. Also mentioned in the Cisco whitepaper (2017) are AWS CloudTrail to track API and log changes to and by functions, and Amazon CloudWatch to log functions invoked with their output, function usage and related metrics. AWS CloudTrail and Amazon CloudWatch provide serverless monitoring, though not exhaustive. AWS provides Amazon VPC (Virtual Private Cloud), in which "you could achieve better security by deploying them within a private networking environment… ensur[ing] that the only way to interact with your data from the Internet will be through the APIs that you've defined and the Lambda code functions that you've written" (Amazon Web Service, 2015). Serverless monitoring is further extended by Cisco's Stealthwatch Cloud, which implements security alert tools from VPC flow logs of serverless AWS Lambda functions (Cisco, 2017).

Aside from AWS and its partner network PureSec, Microsoft Azure Functions has a mere abstract and a link to download a white paper on serverless platform security that has not yet been published (Shinder, 2018). Google Cloud Functions website presents Google Cloud IAM (Identity and Access Management), which allows clients to adopt the security principle of least privilege, but without specific or custom references to be applicable in serverless environments. Google does offer Stackdriver Monitoring, which allows API calls and function executions to be monitored (Google Cloud, 2018). However, gauging how adequate those provisions would be to ensure security of a serverless application is an entire industry of research in itself. IBM Cloud

Functions website only has a security bulletin reporting two function runtime vulnerabilities affecting its serverless functions (IBM, 2018).

Conclusion

Approximately a decade since the cloud revolutionized the world of computing and database, this writer concurs with many authors cited in this paper as well as the general consensus in the web that serverless computing and FaaS is the next big evolution in the world of cloud computing. Serverless offers many attractive benefits such as: faster push time to market an application; ease of design and development by removing server-side out of the equation, allowing even less tech-savvy developers to launch applications; more efficient resource usage by delegating server side management to cloud providers; lower cost of development and operations with pay-per-function options; less maintenance with automated workflows driven by functions or events; no risk of overpaying by over-provisioning server instances; and with auto-scaling, removing the risk of performance detriment caused by under-provisioning. However, serverless carries as many drawbacks as the benefits: pricing may not be as inexpensive as advertised, especially in larger applications; an outside vendor controls your system; serverless vendor’s implementation creates vendor lock-in; developers are at the mercy of the serverless vendor (or options are not available) for server optimization, debugging, monitoring, integration testing, and versioning; and serverless is not optimal for larger or long term applications.

Moreover, perhaps the most significant issue for serverless architectures concerns database security. Serverless design inherently creates security vulnerabilities with: multi-tenancy that raises not only performance but also potential dangers with data storage; a larger attack surface; a more complex attack surface which makes monitoring, debugging and testing more difficult; system complexity that hinders placing a powerful authentication scheme extensively throughout the serverless functions; and a lack of industry accepted best practices or standards on how to securely configure serverless systems. As with all new innovations, serverless database security demands more research and development to keep pace with the rapid proliferation and adoption of serverless computing. This writer also concludes in agreement with majority of the sources cited in this paper that perhaps the best and most prudent option at this point in time is for each user to carefully weigh the advantages and disadvantages of going serverless before diving into the deep unknown.

References

Akiwatkar, R. (2017, May 15). The Drawbacks of Serverless Architecture - DZone Cloud. Retrieved December 4, 2018, from https://dzone.com/articles/the-drawbacks-of-serverless-architecture

Amazon Web Services. (2015). AWS Serverless Multi-Tier Architectures [Whitepaper]. Retrieved December 3, 2018 from file:///D:/SK%20KSU/2018%20Fall/DB%20Sec%20Aud/Serverless%20Paper/Whitepaper_AWS_Serverless_Multi-Tier_Architectures.pdf

Cisco. (2017). Monitoring Serverless Architectures in AWS [Whitepaper]. Retrieved December 3, 2018 from https://www.presidio.com/resources/files/1314/white-paper-c11-739854.pdf

Dwyer, I. (2016). Serverless Computing [Whitepaper]. Retrieved December 3, 2018 from file:///D:/SK%20KSU/2018%20Fall/DB%20Sec%20Aud/Serverless%20Paper/Whitepaper_Serverless_Final_V2.pdf

Eivy, A. (2017). Be Wary of the Economics of "Serverless" Cloud Computing, *IEEE Cloud Computing*, *4(2)*, 6-12. http://doi.ieeecomputersociety.org/10.1109/MCC.2017.32

Google Cloud. (2018, November 7). Monitoring Cloud Functions | Cloud Functions Documentation | Google Cloud. Retrieved December 5, 2018, from https://cloud.google.com/functions/docs/monitoring/

Google Cloud. (2017). What is serverless [Whitepaper]. Retrieved December 3, 2018 from Google Cloud: https://cloud.google.com/serverless/whitepaper/

IBM. (2018, July 29). Security Bulletin: IBM Cloud Functions is affected by two function runtime vulnerabilities. Retrieved December 5, 2018, from http://www-01.ibm.com/support/docview.wss?mhq=cloud functions security&mhsrc=ibmsearch_a&uid=ibm10718977

Lee, H., Satyam, K. & Fox, G. (2018). Evaluation of Production Serverless Computing Environments. 2*018 IEEE 11th International Conference on Cloud Computing*, 442-450. doi: 10.1109/CLOUD.2018.00062

Lynn, T., Rosati, P., Lejeune, A., & Emeakaroha, V. (2017). A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-a-Service) Platforms. *2017 IEEE 9th International Conference on Cloud Computing Technology and Science*, 162-169. doi: 10.1109/CloudCom.2017.15

Patrizio, A. (2018, April 12). One in five serverless apps has a critical security vulnerability. Retrieved December 3, 2018 from https://www.networkworld.com/article/3268415/security/one-in-five-serverless-apps-has-a-critical-security-vulnerability.html

Prothero, D. (n.d.). What is Serverless Architecture? Retrieved November 8, 2018, from https://www.twilio.com/docs/glossary/what-is-serverless-architecture

PureSec. (2018). PureSec Compared with Traditional Application Security Products. Retrieved December 5, 2018, from https://www.puresec.io/ssre_vs_traditional

Roberts, M. (2018, May 18). Serverless Architectures. Retrieved December 3, 2018, from https://martinfowler.com/articles/serverless.html

Rogers, O. (2017). Economics of Serverless Cloud Computing [Whitepaper]. Retrieved December 3, 2018 from file:///D:/SK%20KSU/2018%20Fall/DB%20Sec%20Aud/Serverless%20Paper/MSWhitepaper_Economics_Serverless_Cloud_Computing.pdf

Savage, N. (2018). Going Serverless. *Communications of the ACM,61*(2). doi:DOI:10.1145/3171583

Segal, O., Zin, S., and Shulman, A. (2018). The Ten Most Critical Security Risks in Serverless Architectures [Whitepaper]. Retrieved December 3, 2018 from https://www.puresec.io/hubfs/SAS-Top10-2018/PureSec%20-%20SAS%20Top%2010%20-%202018.pdf

Serverless Computing – Amazon Web Services. (n.d.). Retrieved November 8, 2018, from https://aws.amazon.com/serverless/

Shinder, T. (2018, June 20). Abstract-Azure Functions and Serverless Platform Security. Retrieved December 5, 2018, from https://docs.microsoft.com/en-us/azure/security/abstract-serverless-platform-security

Simi. (2018, October 16). Pros and cons of serverless computing. Retrieved December 3, 2018, from https://www.nine.ch/en/blog/what-is-serverless-computing

Thomas, I. (2018, August 22). Top 10 Security Risks in Serverless Architectures - DZone Security. Retrieved December 4, 2018, from https://dzone.com/articles/top-10-security-risks-in-serverless

Topper, J. (2018). Compliance is not security. *Computer Fraud & Security,2018*(3), 5-8. doi:10.1016/s1361-3723(18)30022-8

Varghese, B., & Buyya, R. (2018). Next generation cloud computing: New trends and research directions. *Future Generation Computer Systems,79*, 849-861. doi:10.1016/j.future.2017.09.020

Watson, M. (2017, May 15). What Is Function-as-a-Service? Serverless Architectures Are Here! Retrieved November 8, 2018, from https://stackify.com/function-as-a-service-serverless-architecture/