T-Mobile Cloud-based Serverless

Security Development

CS547: Secure Systems and Programs

City University of Seattle

Hema Deepika Thanigai Arasu - [thanigaiarasuhemade@cityuniversity.edu](mailto:thanigaiarasuhemade@cityuniversity.edu)

Matthew Thibault - [thibaultmatthew@cityuniversity.edu](mailto:thibaultmatthew@cityuniversity.edu)

Vu Pham – [phamvu@cityuniversity.edu](mailto:phamvu@cityuniversity.edu)

**ABSTRACT**

Serverless is a cloud computing model that allows developers to direct their limited time and effort on business logic and leave cloud computing providers to address any server-related infrastructures and maintenance. Both developers and organizations continue to embrace serverless computing to not only reduce but also eliminate the overhead of infrastructure, scaling, and provisioning. Amazon Web Services (AWS) is a popular service from Amazon company that provides cloud services all over the world. This paper analyzes how T-Mobile company accelerates adoption serverless cloud-based computing with AWS and how it impacts its security development. The paper emphasizes the goals and benefits of serverless computing for company as well as the improvement of development quality and security. The security will be examined bases on the Jazz platform for serverless architecture.

Key words: Serverless, cloud-computing, AWS Lambda, scalability

1. **INTRODUCTION**

T-Mobile US, Inc is a technology company that offers wireless services over a nationwide 4G LTE network to more than 70 million customers. As T-Mobile experimented with Amazon AWS serverless technologies, the service helped cut down on the time spent in development and deployment and also helped cut down the use of other resources. Additionally, serverless can improve the security system while developing software and services. With the benefits that serverless brings to the company, T-Mobile now has the “serverless first” policy for the development of new services. To achieve the goals, the company has come up with Jazz. This platform has helped many development teams have a good approach to using serverless computing. The Jazz platform will be researched in this paper as well as the security improvement serverless provides for the company.

The findings of this project will be useful to computer science students, researchers, and organizations because of the valuable insights that will be shared regarding the use of serverless computing for software development and its impact on security development. Students and researchers around security development will benefit from the knowledge of this project when carrying out research on the topic. It will provide them with useful background about the application of AWS and its role in security development. Organizational leaders, particularly technology leaders will gain valuable insights into the use of AWS in security development and the benefits that they could realize through serverless computing. Unlike other studies which have been general, this project is specific to T-Mobile and its security development operations.

This project relies on Amazon Web Services (AWS), particularly AWS Lambda, which is a revolutionary serverless computing platform that provides the capability to run code without managing a server. The project focuses on how T-Mobile utilizes AWS Lambda for secure development.

1. **LITERATURE REVIEW**

Serverless computing is one of the most revolutionary developments in cloud computing. George and George (2021) defined serverless computing as a cloud computing technique that offers backend services on an as-utilized basis. It allows users to write and deploy codes without thinking about the infrastructure being used. Serverless computing allows developers to invest their time and effort in the application logic rather than the development and management of infrastructure (Venugopal & Reddy, 2021). Developers only need to deploy applications to the serverless cloud as a function and leave the cloud provider to take care of the management, scaling, and provisioning of the resources needed to ensure that the systems run as smoothly as possible (Hassan et al., 2021). The origin of serverless computing can be traced to the introduction of Amazon Lambda in 2014 before being adopted by other cloud computing companies, such as Microsoft and Google in 2016.

There are several defining features of serverless computing. One of them is scalability, which is concerned with the ability to meet demand during both peak hours and downtime (Shafiei et al., 2021). Scaling of resources can be vertical when resources are added or removed from the running container, and horizontal when new containers are created, or excess containers are eliminated without compromising the ability to meet current demand.

Serverless computing allows the automatic scaling up or down of resources based on demand, to allow the developer to focus on the core tasks involved in the development of the application rather than management of resources (Shelar et al., 2021). The second important characteristic of cloud computing is payment per resource usage. Developers are charged based on the number of resources they use at any point in time (Marin et al., 2022). Serverless providers only charge for the resources consumed when the application uses the resources rather than when the application is idle.

The application of serverless computing capabilities promises developers and organizations a wide range of benefits. According to Ajithkumar et al. (2020), the benefits of serverless computing have contributed to its popularity among developers. Benefits such as lack of need for management of infrastructure, minimal costs, and infinite elasticity make serverless computing a preferred choice over other solutions, such as Platform-as-a-Service (PaaS) (Ajithkumar et al., 2020). According to Wen et al. (2020), the fact that serverless computing eliminates the need for focus on the management of infrastructure leaves developers excited, particularly for those who do not have a background in the management of hardware infrastructure. Serverless providers can manage resources in a more unified manner, which improves the utilization of resources and reduces resource waste (Andi, 2021). It is also possible for developers to utilize open source serverless platforms such as OpenFaaS and OpenWhistk. Serverless computing enables developers to build simple functions capable of independently performing a single purpose, such as making an API call (Kulkarni, 2022). The application of serverless computing enables quicker turnaround by significantly cutting time to market. Developers may simply add and modify code on a piecemeal basis rather than following a complicated deployment process of rolling out fixes and new features.

1. **METHODOLOGY**

This paper analyzes how T-Mobile company accelerates the adoption of serverless cloud-based computing with AWS and how it impacts its security development. The study employed a qualitative methodology to find out the application and impact of serverless computing at T-Mobile Company. Qualitative methodology involves the collection and analysis of non-numerical data to develop an in-depth understanding of experiences, opinions, and concepts (Aspers & Corte, 2019). Specifically, the study utilized a qualitative research design, which focuses on the exploration of a phenomenon within some context through the collection and analysis of qualitative data (Rashid & Waseem, 2019). The collection of data involved the use of document analysis, also referred to as document study, where the researchers have reviewed written materials, including policy documents, guidelines, archives, and annual reports relating to T-Mobile and its application of serverless computing. Data was analyzed using thematic analysis, which focused on the identification of patterns and themes.

1. **BACKGROUND OF SERVERLESS COMPUTING**

Serverless computing is a relatively new paradigm in cloud computing. This is a cloud computing model in which a cloud service provider manages the infrastructure and automatically creates and scales computing as needed to run application code in response to events. This allows developers to focus on writing code without worrying about the underlying infrastructure. Instead of provisioning and managing servers, cloud providers handle their tasks so developers can focus on their applications.

The idea of serverless computing is not new. Until now, developers have used Platform as a Service (PaaS) solutions, which abstracts much of the infrastructure management. However, the PaaS required developers to specify the number of servers of virtual machines required to run the applications.

The concept of serverless computing can be traced back to the early 2000s, when utility computing and cloud computing were emerging as new paradigms for delivering computing resources. The term “serverless” became popular with AWS Lambda, which was released in 2014. AWS Lambda allowed developers to run code without provisioning or managing servers. Instead, developers can create functions that run in response to events, such as changes to a database or uploading a file.

There are several companies that provide cloud services such as Microsoft and Google soon followed suit and releasing their own serverless services. Nowadays, serverless computing has become a popular choice for many developers, especially those building microservice-based architectures.

1. **SERVERLESS COMPUTING PROVIDERS**

The three leading serverless providers include Amazon Web Services (AWS), Microsoft Azure, and Google Cloud. Generally, AWS offers more and superior features than Azure and Google. AWS dominates in IaaS services, but it is relative expensive with relatively weak security features when compared to Azure and Google Cloud (Kamal et al. 2020). Microsoft Azure is another leading cloud serverless provide but particularly dominant in SaaS and PaaS services. The features of Azure are relatively better than those of AWS. Google Cloud is a competitor of AWS and Azure, especially because it is relatively cheaper and comes with better security features (Kamal et al. 2020). Each of the three services use SaaS, PaaS, and IaaS deployment models.

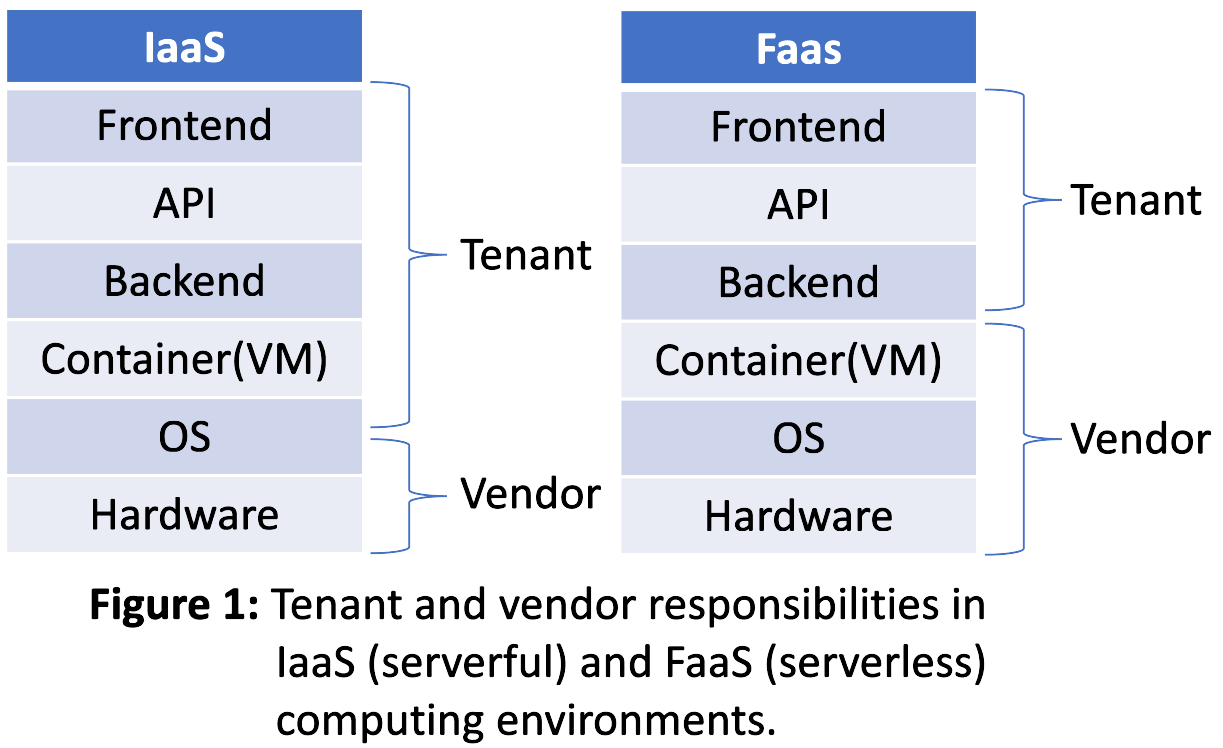
Each of the three serverless providers have dedicated serverless services that can be used by developers to run code without provisioning or managing servers. These are AWS Lambda, Azure Functions, and Google Cloud Functions (Grogan, et al., 2020). Each of the three services allows developers to only pay for the time that the functions are running rather than charging them continuously regardless of whether the cloud server is active. However, the three services have some important differences. First, the three services differ in terms of deployment models (Grogan, et al., 2020).. In AWS Lambda, all functions are deployed in the Lambda environment on servers running Amazon Linux.

While it is possible for Lambda functions to interact with other services, their deployment is limited to the Lambda service. For Google Cloud Functions, functions must be stored in Google’s Container Registry and functions can be executed only as containers (Grogan, et al., 2020). Unlike AWS Lambda and Google Cloud Functions, Azure Functions is more complex and flexible regarding the deployment of serverless functions.

The three services also differ in terms of programming language support. While all three services are capable of directly executing serverless functions in Python and Java, only Azure supports TypeScript and JavaScript, and only AWS Lambda and Google Cloud Functions support Ruby, Node.js, and Go (Arkhipov, 2021). Only AWS Lambda and Azure Functions support C# and PowerShell and only Google Cloud Services supports PHP. However, the addition of an abstraction layer that allows a service to handle codes written in a language that is not natively supported enables AWS Lambda and Azure Functions to use any of the programming languages (Scheuner & Leitner, 2022). Google Cloud Functions only handles codes written in natively supported languages.

1. **SERVERLESS SECURITY CHALLENGES**

As discussed earlier, serverless computing (function-as-a-service, FaaS) removes the overhead of infrastructure management from the development team and outsources these responsibilities to the cloud provider, including the security of said infrastructure as well as system hypervisors and virtual machines (VMs). Meanwhile the tenants of the serverless platform retain the security and privacy responsibilities of resources, workloads, and data (Hong, 2018). This contrasts with serverful computing (Infrastructure-as-a-service, IaaS) in which the infrastructure is maintained by the provider, but VM control and management as well as resource allocation to VMs is the responsibility of the tenant. This alteration in the division of responsibility, depicted in figure 1, offloads some of the tenants’ security concerns, namely any concerns involving physical or on-site attacks (which both platforms handle) as well as any form of exploit designed to target the serverless architecture, tools, VMs, and operating systems. Consequently, the tenant instead may focus security efforts solely on the applications that they have hosted in the cloud and the front ends facing their clients.



Both Faas and Iaas are examples of cloud computing environments meaning that they are environments hosted on a vendor’s servers, and both differ from an even older strategy of computing known as on-premises. With on-premises computing the server infrastructure that houses a business’s applications is owned, operated, and maintained by the business itself. This paper is focused on the similarities and differences of FaaS and IaaS and will not go into the details of on-premises computing. It is worth noting that on-premises, serverless computing exists as well but it is the exception rather than the rule.

Given that serverless and serverful computing are both cloud computing environments it should come as no surprise that they share many of the same security and privacy challenges. These similarities include dynamic baselining, third party dependency management, access control, and API testing and security (Koptyev, 2019). Dynamic baselining refers to the fact that the IaaS containers and FaaS functions can vary wildly over time, which means that tools used to aid security tasks in these environments must be able to handle such dynamism. Importing third-party code is normal in both FaaS and IaaS and the security of any such code is important. Any third-party code incorporated into a FaaS of IaaS should be vetted and regularly updated to maintain the integrity of the environment. While not trivial, it is nonetheless possible for compromised containers or serverless functions to affect other containers or serverless functions operating on the same infrastructure. Careful access control management is the key to keeping separate instances from crashing each other. Finally, APIs are critical to both types of computing environments and should be thoroughly secured.

While FaaS and IaaS have their similarities, serverless computing does have its share of unique security challenges. FaaS generally comes with the downside of exposing more of an application’s functions to the outside. In turn this can make it challenging to implement effective security at the perimeter and relies much more on security between functions with a magnified focus on concepts such as least privilege, separation of privileges, and least common mechanism.

Additionally, although the security of the serverless architecture and services fall under the purview of the serverless provider, proper implementation of those services is the domain of the tenant and therefore adds an additional gap that could be exploited. Managing proper permissions for serverless functions can be a complicated business that can result in vulnerabilities. The difficulty in managing permissions stems in part from the highly adaptable services that serverless offers to be as accessible as possible to a wide range of potential tenants (Sharaf, 2020).

1. **REMEDIES FOR SECURITY CHALLENGES OF SERVERLESS COMPUTING**

Sharaf (2020) identifies three key solutions to the problem of security in serverless computing. The first involves controlling the permission to manage, which helps reduce the danger of an independent function that contains a malicious security threat. This is because the independent functions involved in a serverless architecture with more permission than they need can present a threat if they are compromised by underlying malware (Sharaf, 2020).

It is important to ensure that every function is reviewed to establish its requirements, guided by the principle of least privilege, and continuously scanning for possible suspicious activity. The second solution involves the application of the security perimeter for every function. The essence of perimeter security for every function is to mitigate the risk of multiple vulnerability points resulting from the serverless application (Sharaf, 2020). Each function should be viewed as a vulnerability point, which necessitates the securing of every function to prevent the spreading of security threats across cloud storage because of a security lapse in a single independent function. The third key solution involves limiting third-party packages with a lot of dependencies to help mitigate third-party dependency risks (Sharaf, 2020). Serverless architecture grants third-parties access to the entire cloud-based environment, which implies that the complexity involved in the implementation of perimeter security for every function can be reduced when dependencies in third-party packages are limited.

1. **CASE SCENARIOS**

As T-Mobile experimented with serverless Amazon Web Services (AWS) technologies for its APIs, microservices, and time and event-based processing, it quickly discovered new value in optimized resources, easier scaling, reduced patching, and timesaving. This allowed the company to focus more on exploration and innovation, resulting in greater agility in responding to customer demands. As a result, T-Mobile now has a “serverless first” policy for developing new services with serverless technologies such as AWS Lambda being used for the company’s websites and the T-Mobile smartphone app. It plays a key role in critical T-Mobile applications.

To get to this point, T-Mobile needs a way for multiple development teams to implement and use serverless computing while also complying with security, operational, deployment and cost requirements of the entire enterprise.

With serverless, the hope is that developers can just focus on code. However, to get the code to production, securely and at scale, there are still many non-coding tasks and topics that must be addressed. Letting the company’s teams use bespoke approaches could have created problems with service crawl, cost overruns, security, and compliance. To come up with a solution, the T-Mobile Cloud CoE team created Jazz, an open-source platform for building, deploying, and debugging cloud-based APIs, function, and static websites on serverless architecture. The platform itself runs on serverless architecture using Amazon Kinesis, which can cost-effectively handle streaming data at any scale; AWS CloudFormation, which uses plain text files to automate and secure the provisioning of applications resources; AWS Lambda, a serverless service that runs code in response to events; and Amazon OpenSearch service, which makes it easy to deploy, secure, use and scale OpenSearch and Elasticsearch for log analytics, full-text search and application monitoring.

The most important benefits of Jazz and serverless computing that T-Mobile achieves is the substantial amount of time development teams can now save. By providing templates and direct feedback within the platform, Jazz helps T-Mobile achieve this speed while avoiding the inconsistencies and human errors that can happen with bespoke architectures. The result is more stable applications and improved customer experiences. The acceleration of development processes enables T-Mobile to innovate faster and build greater value for customers.

With the time-consuming development and deployment steps sped up, developers can focus more on security checking, innovating, and iterating on the solutions that customers demand. Additionally, the company can save money with serverless computing by bringing new services to the market faster, and by reducing and optimizing its use of resources.

1. **CONCLUSION**

Serverless computing is an important development for writing code. The use of serverless computing eliminates the need to focus on the management of infrastructure while allowing developers to simply add and modify code on a piecemeal basis. One of the most employed serverless computing platforms is Jazz, which allows the seamless creation, deployment, management, monitoring, and debugging of cloud native applications while leveraging the capabilities of serverless computing. The adoption of Jazz at T-Mobile helped achieve stability in applications while improving customer experiences. T-Mobile has been successful in innovating and quickly introducing new services to the market. The security of serverless computing is an important concern for T-Mobile. The three key solutions to the problem of security include controlling the permission to manage, application of the security perimeter for every function, and limiting third-party packages with a lot of dependencies.

1. **REFERENCES**

Ajithkumar, S., Raghavan, A., & Nasreen, A. (2020). Serverless computing a cloud technology revamp. *International Research Journal of Engineering and Technology (IRJET), 7*(5), 3153-3156.

Andi, H. K. (2021). Analysis of serverless computing techniques in cloud software framework. *Journal of ISMAC, 3*(3), 221-234.

Arkhipov, A. (2021, February 1). *AWS Lambda vs Google Cloud Functions vs Azure Functions — What to choose in 2023?* Retrieved from TechMagic: https://www.techmagic.co/blog/aws-lambda-vs-google-cloud-functions-vs-azure-functions/

Aspers, P., & Corte, U. (2019). What is qualitative in qualitative research. *Qualitative Sociology, 42*(2), 1-22.

George, A. S., & George, A. S. (2021). Server less computing: the next stage in cloud computing’s evolution and an empowerment of a new generation of developers. *International Journal of All Research Education and Scientific Methods (IJARESM), 9*(4), 21-35.

Grogan, J., Mulready, C., McDermott, J., Urbanavicius, M., Yilmaz, M., Abgaz, Y., . . . Clarke, P. (2020). A Multivocal literature review of Function-as-a-Service (FaaS) infrastructures and implications for software developers. *Systems, Software and Services Process Improvement - 27th European Conference, EuroSPI 2020 Proceedings* (pp. 58-75). Springer.

Hassan, H. B., Barakat, S. A., & Sarhan, Q. I. (2021). Survey on serverless computing. *Journal of Cloud Computing, 10*(39). doi:10.1186/s13677-021-00253-7

Hong, S., Srivastava, A., Shambrook, W., Dumitras, T. (2018, July 9) Go serverless: Securing cloud via serverless design patterns. [Conference session] HotCloud 18’, Boston, MA, United States. https://www.usenix.org/system/files/conference/hotcloud18/hotcloud18-paper-hong.pdf

Himanshu Pant. (2018, March 29). *A brief history of serverless (or, how I learned to stop worrying and start loving the cloud)*. <https://www.freecodecamp.org/news/a-brief-history-of-serverless-or-how-i-learned-to-stop-worrying-and-start-loving-the-cloud-7e2fc633310d/>

Kamal, M. A., Raza, H. W., Alam, M. M., & Mohd Su’ud, M. (2020). Highlight the features of AWS, GCP and Microsoft Azure that have an impact when choosing a cloud service provider. *International Journal of Recent Technology and Engineering (IJRTE), 8*(5), 4124-4132.

Koptyev, S. (2019, April 12). *Key differences in security, management for serverless vs. containers* TheNewStack.

<https://thenewstack.io/key-differences-in-security-management-for-serverless-vs-containers/>

Kulkarni, V. (2022). A research paper on serverless computing. *International Journal of Engineering Research and Technology, 11*(9), 351-353.

Malireddi, S. (n.d.). Accelerate Companywide

Adoption of AWS Serverless Computing with T-Mobile | Case Study | AWS. Amazon Web Services, Inc. https://aws.amazon.com/solutions/case-studies/tmobile/

Marin, E., Perino, D., & Pietro, R. D. (2022).

Serverless computing: A security perspective. *Journal of Cloud Computing, 11*(69). doi:10.1186/s13677-022-00347-w

Rashid, Y. R., & Waseem, A. (2019). Case study method: A step-by-step guide for business researchers. *International Journal of Qualitative Methods, 18*(1), 1-12.

Roberts, M. (n.d.). Serverless Architectures. martinfowler.com. <https://martinfowler.com/articles/serverless.htm>

Retter, M. (2021, November 18). History Of Serverless Computing: Where It Started. Dashbird. <https://dashbird.io/blog/origin-of-serverless/>

Scheuner, J., & Leitner, P. (2022). Function-as-a-Service performance evaluation: A multivocal literature review. *Journal of Systems and Software (JSS), 1*(1), 1-24.

Shafiei, H., Khonzari, A., & Mousavi, P. (2021). Serverless computing: A survey of opportunities, challenges, and applications. *Association for Computing Machinery, 1*(1), 1-23.

Sharaf, S. (2020). Security issues in serverless computing architecture. *International Journal of Emerging Trends in Engineering Research, 8*(2), 539-544. <https://doi.org/10.30534/ijeter/2020/43822020>

Shelar, N., Kumbhare, S., & Gorde, A. (2021). Build a serverless website using AWS cloud. *International Journal of Creative Research Thoughts (IJCRT), 9*(7), 868-873.

Venugopal, M. V., & Reddy, C. R. (2021). Serverless Through Cloud Native Architecture. *International Journal of Engineering Research & Technology (IJERT), 10*(2), 484-496.

Wen, J., Chen, Z., Jin, X., & Liu, X. (2020). Rise of the planet of serverless computing: A systematic review. *Manuscript submitted to ACM* (pp. 1-60). Association for Computing Machinery.