8-2022

TAXONOMY OF SECURITY AND PRIVACY ISSUES IN SERVERLESS COMPUTING

Vasanta Swarna Ratnam Pusuluri

Taxonomy Of Security and Privacy Issues in Serverless Computing

by

Vasanta Swarna Ratnam Pusuluri

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree

Master of Science in

Information Assurance

June, 2022

Starred Paper Committee:
Abdullah Abu Hussein, Chairperson
Lynn Collen
Akalanka Mailewa

Abstract

The advent of cloud computing has led to a new era of computer usage. Networking, and physical security are some of the IT infrastructure concerns that IT administrators around the world had to worry about for their individual environments. Cloud computing took away that burden and redefined the meaning of IT administrators. Serverless computing as it relates to secure software development is creating the same kind of change. Developers can quickly spin up a secure development environment in a matter of minutes without having to worry about any of the underlying infrastructure setups. In the paper, we will look at the merits and demerits of serverless computing, what is drawing the demand for serverless computing among developers, the security and privacy issues of serverless technology, and detail the parameters to consider when setting up and using a secure development environment based on serverless computing.

Table of Contents

Page

List of Figures 5

Chapter

I. Introduction 6

Introduction 6

Problem Statement 7

Nature and Significance of the Problem 8

Objective of the Study 8

Study Questions/Hypotheses 8

Definition of Terms 9

Summary 10

II. Background and Review of Literature 11

Introduction 11

Background Related to the Problem 11

Characteristics of Serverless Computing 12

Literature Related to the Problem 15

Literature Related to the Methodology 18

Summary 24

III. Methodology 25

Introduction 25

Design of the Study 25

Chapter Page

Data Collection 27

IV. Data Presentation and Analysis 31

Introduction 31

Data Presentation 31

Data Analysis 95

Summary 95

V. Results, Conclusion, and Recommendations 96

Introduction 96

Results 96

Conclusion 98

Future Work 98

References 100

List of Figures

Figure Page

1. Serverless Architecture 12

2. Data Collection 28

3. Analyzing and Interpreting the Data 29

4. Classification of the Data 30

5. Taxonomy of Serverless Computing 32

6. Taxonomy of Development Tools 66

Chapter I: Introduction

Introduction

There was a time where cloud computing was so popular among everyone in the IT field. It gave an opportunity to abstract the physical hosting environment. Later, the hosting environments have been scaled down by hosting different hardware units. Since the beginning of another era in the IT industry with the concept of virtualization, virtual machines in the cloud have been put in service. Then cloud services increased their intensity by providing a platform as a service (PAAS). The Operating systems with run time shared within the cloud. Then vendors started sharing software and applications within the cloud and the users no longer had to worry about the infrastructure and any underutilized resources. Without noticing people have stepped into the serverless era of cloud computing. Now the function-based scaling was reached which is also referred to as the serverless architecture.

Serverless computing is an emerging technology and became a controversial topic in no time. It allows to build, and run the applications besides worrying about the servers, thereby providing seamless hosting and execution environment. But it is still a new technology, and the developers are trying to get familiar with its functionality and security features. Hence, this paper discusses these topics to make the developers understand their role and responsibility security-wise when they work in serverless computing to keep the serverless environment safe from attackers.

This paper provides a state-of-the-art survey that dives deep into serverless computing with intention of providing a piece of sound knowledge to the reader to educate themselves with its pros and cons on serverless technology. Starting from the serverless architecture, the

discussion will be continued about its attributes by putting more weight on security and privacy issues. This paper distinguishes serverless architecture from the traditional cloud architecture and explores why developers would want to use serverless computing in their applications and how it affects software development. Furthermore, serverless enabling technologies will be introduced along with their characteristics. Later, the contemporary security and privacy issues related to serverless computing will be discussed and will be specified if there are any security features implemented to mitigate security risks by defending against these security and privacy issues.

Problem Statement

Serverless computing aids in running the applications without worrying about the servers. However, the underlying infrastructure like servers belongs to third parties. So, the developers do not have full control of the system and its data flow which results in security and privacy issues. The prevailing solutions are designed based upon the traditional vulnerabilities, but the security evolves whiles the attackers keep shifting their strategies. Hence, we are talking about the problem that is different from the classical application. The serverless technology relies on Functional as a Service (Faas) and the functions concerning the application are available to the public. It is running on the shared platforms, which creates a potential way for attackers to perform malicious activities. The solutions for the serverless security attacks are derived using static analyzing functions which restricts to few policies. This results in increasing the attack surface. Also, there is very little monitoring in serverless applications. Hence tracking of all the attacks is a big concern. This is a new technology that is more suspectable to zero-day attacks. Some of the subproblems are Overprivileged Function Permissions, using third party dependencies that are not secure, and Cross-Execution Data Persistency

Nature and Significance of the Problem

According to the NEWSTACK 2018, more than 75% of the developers and users plan or to apply serverless computing as a business solution (Rice, 2017). There is no assurance from the providers to ensure security. With all the security flaws, it makes the system vulnerable. *Exampl*e: AWS Lambda which is a serverless service was once pruned to DDOS attack This, in turn, led to outages to many different services that use the same public DNS (Rice, 2017). The breach that happened on Cloudflare which was named Cloud bleed caused a memory leak of their customer's data (Conger, 2017). Hence the company has requested all its customers to rotate their passwords.

An intruder gained access to 100 million Americans and 6 million Canadians personal information of Capital One credit card customers that store their data in AWS (Sheridan, 2019). The Pursec audit states that their serverless projects have one or more critical vulnerabilities or misconfigurations (Ltd, 2018a, b). Few of the projects share confidential data such as the API keys and the credentials through publicly available code repositories. Around 1,00,000 GitHub repos have leaked their API tokens and cryptographic keys (Catalin, 2019).

Objective of the Study

The objective of the study is to educate users and developers on the security and privacy issues in serverless computing. This study also provides the mitigation techniques for the identified concerns, thereby enabling serverless technologies.

Study Questions/Hypotheses

1. What are the Security and privacy issues concerning serverless technology?
2. How to detect the vulnerabilities in serverless computing?

3. How to respond to the incidents?
4. What to protect while using serverless?
5. What are the current trends and open problems?
6. How users should protect themselves while using serverless computing

Definition of Terms

FAAS–Function as a Service–The Function as a service allows their customers to build, run and maintain the applications without worrying about the complexity, infrastructure. An example of this service is serverless. The services are maintained by third party API'S.

Serverless Computing–The Serverless computing allows the developers to develop and build the applications without worrying about the servers and the maintenance. The serverless vendors take care of the servers.

PAAS–Platform as a Service–This service offers software and hardware over the internet. Application and the data are managed by the customers.

Cloud Computing–The computer services that are available to the customers without managing the resources. All the services in the cloud are delivered through the internet

IAAS–Internet as a Service–This is typically a cloud-based service, in which the computer, storage, and the network can be accessed using the internet. Application, data, OS is managed by the customers.

IT–Information Technology–It is the study of computer, networks, and telecommunication which focuses on accessing, storing, managing and sending the data.

API–Application Programming Interface–The intermediate software that connects different applications to communicate with each other through this interface.

Zero-day Attacks–The attack that is performed before the weakness on the system is discovered and the solution for the vulnerability is being identified.

Cloud Foundry (multi-code application platforms)–IBM uses cloud foundry to build and deploy code, while they coordinate with the given services which ensure fast deployment of cloud-based applications (*IBM Cloud Foundry–Overview*, 2021).

AWS–Amazon Web Services–Amazon launched reliable and inexpensive services which are named AWS. This web service allows pay-as-you-go service which attracted most of the developers to build their applications. (*Amazon web services (AWS)*, 2021).

DDoS–Distributed Denial of Service Attack–The Attack that disrupts the normal traffic which makes services become unavailable (*What is a DDoS attack?*, 2020).

Malware–This is software that damages the system, network, or gains unauthorized access to a PC.

SQL Injection–This attack occurs by injecting malicious SQL statements there by gaining access to the application, modify data, and disclosure the data.

Cross-Site Scripting (XSS)–This attack occurs by injecting malicious scripts into the websites and access users' data by gaining full control of the system.

Summary

This chapter has covered the introduction to our research on serverless computing. We have discussed the problems concerning serverless technologies and the nature of the problem by providing the security threats that have happened in the real world. We have also discussed the hypothesis on which we are going to work in the future chapters.

Chapter II: Background and Review of Literature

Introduction

This chapter gives in-depth knowledge on serverless computing security and privacy issues. All the security concerns that were faced by this technology will be elaborated in depth by providing real-time events. Later, the background related to the problem and the architecture will be discussed.

Background Related to the Problem

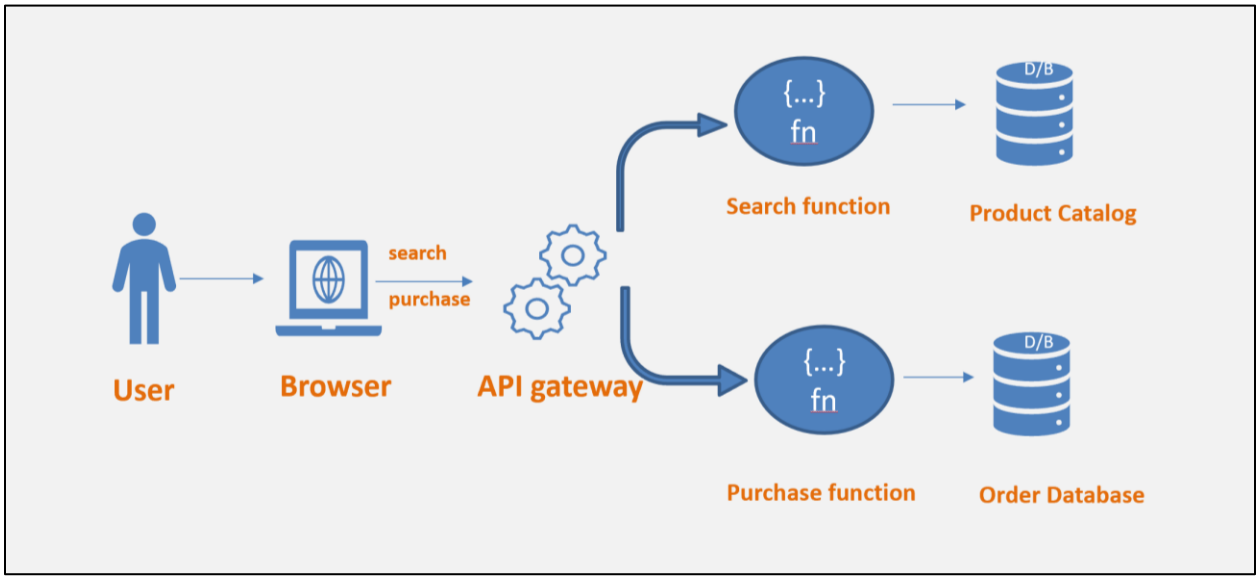
Earlier, system admins would take care of all the memory allocation, servers, drivers, upgrades, installations, etc. This process is known as the “Bare-Metal” environment (Retter, 2021). Then, Virtual machines came into the spotlight. This made developer's life so easy by enabling them to switch from one server to another. Later, containerization technologies were used. This enables multiple applications to run on the same system without interfering with each other. Finally, serverless came into existence which was developed by Austen Collins and introduced in October 2015 (Serverless Framework–AWS lambada guide, 2021). Amazon AWS lambda first builds the serverless framework using Node.js. This relies on Function as a service (FAAS). Small chunks of code or functions are taken as inputs, processes, return the results and shutdown. The cloud providers take care of the Server or cluster provisioning, Patching, Operating system maintenance, Capacity provisioning,

Administering servers for backend components (compute, databases, storage, stream processing, message queueing, etc.). Developers can build the system using serverless without worrying about the infrastructure, maintenance, servers, hardware, etc. After serverless was introduced, many other providers came up with varying processes, features that can support

multiple technologies. It has the auto-scaling service that makes this technology easier to maintain and build. Since it is a new and emerging technology, there are few issues that make serverless vulnerable. The figure below depicts how serverless technology works

Figure 1

Serverless Architecture



The popular serverless providers are Amazon AWS Lambda, Google cloud functions, Microsoft Azure, Apache open whisk, Tencent SCF, IBM cloud functions, Cloudflare Workers, Kubeless, Knative, Alibaba cloud, Spotinst, Fn (*Serverless infrastructure providers*, 2020).

Characteristics of Serverless Computing

Below are the various characteristics that define serverless computing.

1. Stateless: Stateless is a typical characteristic of serverless computing. Using Function as a service (FAAS), nothing is stored in the memory as the code is automatically created and deleted by our own platform. So, by this more instances can be signed up.

The only disadvantage in this characteristic is that one will not be able to use HTTP sessions.

2. Efficient: Serverless enables efficiency by allowing the developers to pay for the resources that are being used. It follows the pay-as-you-go billing model.
3. Auto-Scaling: Resources made available as the request comes in and is managed automatically. For example, Amazon Aurora Serverless provides auto-scaling, on-demand by automatically starting and shutting down the databases and scale capacity based on application needs (Amazon Aurora Serverless, 2021).
4. Security: Security is a big concern nowadays, however serverless plays its best to provide security in all aspects. Serverless considers all the vulnerabilities and follows the best practices like regular patching, adopting principles of least privileges, fumigate all the inputs, monitor, and log functions. Also, as the memory is not stored it has a very low risk of long-term attacks. For example, in AWS Lambda all the communications were encrypted with Transport Layer Security.
5. Debugging: Serverless supports debugging for their developers by enabling them to find the errors and bottlenecks.
6. Programming languages: Serverless supports many programming languages for their developers. They can choose a language based upon their choice to build any applications in the serverless platform. For example, AWS supports C#, Java, Python, Node.js, Ruby, PowerShell, and Go (Heath, 2019). They also enable runtime API which makes them to use additional programming language.

7. Composability: Functions in one serverless can be invoked from others which makes easy computation and development of complex serverless applications.
8. Hostless: With the serverless architecture, users need not worry about the servers, upgrades, or security patches. This reduces the operational costs. They can simply install and run the application with little knowledge of the application configurations (H, 2019).

As mentioned above, serverless works with Functions as a Service (FaaS). First, the developers or the users write the function to provide a precise objective. Then, he defines the event to trigger the cloud service. The cloud service provider verifies if the function instance is being used or not. If it is not used, the new function starts. The user can see the executed function results inside the application (Johnson, 2019).

As the coin has two sides, there are certain limitations with serverless computing.

1. Lack of Control: The serverless is provided by third-party API's. The configurations are limited, and the users will not be exposed to different configurations. Also, if an issue occurs other than the one in code and configurations, the user will not have control over the issues, and it can only be resolved by the platform provider.
2. Security: There is no surprise that serverless technology has vulnerabilities as it is very new and can be prone to security issues. Firstly, it uses different providers across multiple regions. This compromises security. Secondly, this new technology is susceptible to a new type of risk such as security orchestration challenges and perimeter fragmentation (Sollow, 2020). All these factors compromise the security.

3. Debugging and testing: It is difficult to debug and test complex applications because of the infrastructure and distributed platform. It might be possible to use third-party tools, but the users might face integrations and security issues (Roberts & Chapin, 2021b).
4. Performance: Due to its abstraction, i.e., hiding of code execution details, serverless tend to show poor performance.
5. Vendor Lock-In: The API acts as an interface between customers and serverless functions and there is a possibility of lock-in happening at this level. The vendor lock-in makes users dependable on the service providers. This makes it difficult for them to migrate from one application to another as the architecture needs to be remodeled (Ltd, 2018a, b).

Literature Related to the Problem

According to the dataset of Synk, most of the data breaches occurred due to the common vulnerabilities.

AWS Lambda was once pruned to DDOS attack which was persistent for 8 hours and the services went unavailable from 10:30 AM to 6:30 PM PDT in 2019. This, in turn, led to outages to many different services that use the same public DNS (Dark Reading Staff, 2019). All the customers were informed that the outage occurred due to a DDoS attack. Currently, Amazon stared using shield advance to protect its services from such attacks. Google Cloud which also uses the serverless platform has reported a DDoS attack in September 2017, causing 2.5 Tbps in traffic. It was lasted for 6 months and targeted thousands of their IP addresses. The attack did not cause any impact, but this leads to the disclose of many vulnerable servers (Kovas, 2020).

The most popular OTT platform Netflix uses serverless architecture and runs using AWS Lambda. Even the coco-cola vending machines use AWS serverless framework. The Distributed Denial of Service (DDOS) attacks resulted in downtime of Netflix servers. *New York Times* which also uses serverless architecture was affected by this attack. This resulted in users not being able to use their websites (Duc, 2019).

Cloudflare which uses serverless functions, once reported that the vulnerability in their website caused a data leak which was named as Cloud Bleed attack that occurred on September 22, 2016. This resulted in the leaking of customer messages, passwords, hotel bookings, etc. This impacted the companies like Fitbit, uber, OK Cupid, and around 3,400 websites that use Cloudflare software (Ahmed, 2017). Cloudflare has requested all its customers to rotate their passwords (Baisakhiya, 2017).

Puresec is designated as the security partner of AWS lambda. The Pursec audit was conducted on open-source serverless projects and it stated that 21% of those projects have one or more critical vulnerabilities or misconfigurations (Ltd, 2018a, b). Six percent of the projects have application secrets that lead to sharing of confidential data such as the API keys and the credentials through the publicly available code repositories. It also stated that the software developers should be aware of the security risks and should gain knowledge on how to protect their applications while building with serverless. This is stated because it reported that those vulnerabilities and weaknesses were due to poor security practices.

One of the vulnerabilities in the AWS lambda function is that the Identity and access management (IAM) is not versioned through the current Lambda functions. This increases the usage of numerous versions of the identical function and causes trouble to add or remove

permissions. The recent incident on Amazon AWS services happened where the credentials are being stolen. It was found that TeamTNT access the docker containers to install malware which is known as crypto-mining malware. This malware botnet steals the credentials and also affects multiple servers by deploying more crypto-miners (Cimpanu, 2021).

Capital One has suffered from a data breach on July 29, 2019, where the intruder gained access to 6 million Canadians and 100 million Americans personal information of their credit card customers. Around 1,40,000 of their social security numbers and 80,000 of the bank accounts have also been compromised. All the data that has been leaked was stored in the AWS and the reports state that the vulnerability in the AWS has caused the data leaks. Around $150 million incremental cost has been experienced by Capital One due to this data breach (Lu, 2019).

According to Zhang et al. (2019), serverless applications can sometimes be over-headed as they are stored remotely. Hence, Shredder is used to interact with the data directly within the storage nodes. But there was a speculative execution attack that rooted data leaks. This occurred due to the side-channel attack that caused to leak victim's confidential data. Not only the manufactures but also the run-time developers worked to mitigate the attacks. Shredders are prone to this type of attack (confidentiality risks) even if the mitigation strategies are being imposed on them.

As a part of the study, the research implemented a code-injection attack on AWS lambda. The hackers were able to leak about 170 bits/second (Alpernas et al., 2018) and proved that the termination channels in serverless computing lead to the security breach.

In addition to this, there are many companies that depend on AWS to perform certain operations. For example, The OneLogin company was hacked on May 31, 2017, at 2 am PST

where the customer database was being accessed by the hackers. This reveals not only the sensitive information about their customers, but also creates a pathway for the hackers to access a set of AWS keys and which makes them to access AWS API. This also affected 2000 companies and 300 app vendors that use OneLogin as a single sign-on for cloud applications (*OneLogin: Breach exposed ability to decrypt data,* 2017).

Literature Related to the Methodology

The paper Maissen et al. (2020) discusses the main providers of serverless computing and made a detailed report on their offerings as a cloud provider. According to their research, AWS lambda was the best of all the serverless providers as it has consistency in reliability and performance. But it lacks in certain features such as some security measures which require changing of API's frequently, i.e., in about 30 seconds which reduces the productivity for developers and testing functions. Microsoft azure functions performance is less compared to AWS lambda. It is not suitable for short sessions, as it takes 2 to 4 seconds for the cold start latency. It uses supplementary tools for local debugging. The Google cloud providers have a well-structured web portal and proper command-line interface tool. This has the higher cold start latency which is higher compared to AWS and IBM cloud functions. But there are few drawbacks such as the deployers have only limited configuration options. The monitoring tool lags in time which prevents them from prompt involvement. IBM Cloud functions are available to the public for free. It helps to perform quick tests by invoking the functions. This contains less waiting time. The main drawback for the cloud function is that it has low performance as the support for the runtime environment is very low and the cloud foundry (multi-code application platforms) supports only willful regions.

According to the blog written by Elena (User, 2021), serverless applications run on event-triggered stateless containers. Developers take care of code, data in cloud and transit, application logic, and configurations whereas serverless providers take care of servers, networks, data centers, storage, containers, OS, and their configurations. As the serverless is provided by third parties, there is no guarantee of security. If the code is not proper, it can be prone to Denial of service (DOS), Cross-site scripting, SQL injection, Broken authorization, and authentication, etc. The outdated third-party libraries cause a pathway for the attackers to hack into the system. The usage of APIs, cloud storage, HTTPS can increase the attack surface. The customization option provided by the serverless can lead to vulnerabilities if they are not set properly. She proposed the techniques to bypass the security issues by encrypting all the passwords used, using key vaults to secure sensitive information. Improper use of APIs needs to be eliminated and HTTP requests need to be validated. Static, Dynamic, and interactive security testing techniques need to be applied to identify weaknesses. Using monitoring tools like rook out, IO pipe helps to identify performance and monitor logs.

According to Jegan et al. (2020), serverless applications can be protected using SecLambda as existing solutions do not have enough security. Basically, serverless applications are divided into small functions and those functions perform specific tasks. Function instance is called when there are requests to process them. Once the request is being handled, the function is paused, and the users pay only for the used sources. As the security challenges increases, log-based anomaly detection tools and traditional Vulnerability scanning tools are not sufficient to overcome the security concerns. In SecLambda, a function runs in a modified container which makes the current function to be in guard mode. This Guard contains a security function

depending upon the function states. According to Puresec (*Cloud workload protection platform. Prisma,* 2020), one vulnerable function can lead to virtual crypto mining without being identified by the user. The SecLambda protects serverless applications by supporting the security policies. This mainly consists of the guard, function runtime, and controller. The controller manages the security functions whereas the guard executes those security functions based on user security policies. This SecLambda helps to prevent data leaks, injection attacks, DOS attacks.

According to Jindal et al. (2021) Serverless applications run based upon the Function-as-a-service, hence the applications are divided in the form of simple functions and are uploaded to the FAAS platform. Those functions are stateless. They are only invoked when the user sends the HTTP request in the FAAS platform. The FAAS platform then helps to deploy and promote the resources to the application functions. All the serverless providers have the FAAS platforms and some are available in the form of Kubernetes. This paper talks about the Apache open whisk, Google cloud platform, and OpenFass. But there are many other platforms that use serverless architecture such as Amazon AWS Lambda, Tencent SCF, Microsoft Azure, IBM cloud functions, Cloudflare Workers, Kubeless, etc.

The research paper Baldini et al. (2017) describes the open problems and current trends of serverless computing. The good thing about serverless is that it scales to zero and will not charge for the inactive time. But this in turn causes cold starts which require some time when invoked for running up the serverless applications. This makes functions longer time than usual to execute. As the functions run on the shared platforms, isolation of those is critical. To guarantee scalability of the functions, it supplies responses to the load and anticipates future load. This can be challenging as the decisions are made with little application knowledge. Also,

multiple serverless platforms and services need to work together with an increase in popularity and there is no guarantee that all the use cases will work in those platforms. Only the developers have control over the supporting tools and what code is deployed. Serverless have limited execution time while some programs might require more execution times. Serverless is stateless, but some real application requires state. Hence, it would be difficult to maintain a state for such applications. This paper discusses some open problems in serverless such as scope i.e., whether it is restricted to the Function-as-a-service, or can it also include other models? In addition to the stateless functions, can serverless computing deal with the state? Currently, the serverless code runs on traditional data centers. Will it also be able to run outside those data centers?

The paper Shafiei et al. (2019) discusses the application of serverless technology in the real world. The stateless nature of serverless makes it suitable for real-time collaboration tools like chatbots, instant messaging tools, real-time tracking. This is used for data analytics where real-time data is streamed into serverless service. The large data can be handled with its auto-scaling feature. Stream Alert is an intrusion detection tool built using AWS Lambda. Serverless architecture automatically secures sensitive information in the public cloud. This also supports IoT services like Kappa platforms that support parallel computing. The serverless technology supports scientific computing such as RNA, DNA computing. The video processing frameworks like Sprocket uses serverless technology. This results in lower cost and lower latency. The pay-as-you-go makes industrial services like gas and oil field management systems use serverless. Hence, this paper proves that serverless is used almost in all fields.

Serverless computing is a way of providing backend services, a platform that hides server usage from developers. The platform is provided by a cloud provider who runs the server and

manages dynamically the allocation of machine resources. The architecture of Serverless allows users to write and deploy code without the bother of worrying about the underlying infrastructure. In the early days of the web, anyone who wanted to build a web application had to have his own physical hardware in other to run a server which was expensive and required a lot of work. The term 'serverless', meaning not using servers can be referred to as peer-to-peer (P2P) software or client-side only solutions. In the cloud context, the current serverless landscape can be said to have come about during an Amazon Web Service (AWS) reinvent event in 2014 and from then, multiple cloud providers, industrial, and academic institutions have come about with their own serverless platforms. Serverless appears to be the natural progression following recent advancements and adoption of Virtual Machine (VM) and container technologies, where each step up the notion layers led to more lightweight units of computation in terms of resource consumption, cost, and speed of development and deployment. Serverless builds upon long-running trends and advances in both distributed systems, publish-subscribe systems, and event-driven programming models, including actor models, reactive programming, and active database systems. Serverless removes infrastructure management responsibilities such as a server or cluster provisioning, patching, operating system maintenance, and capacity provisioning. You can build them for almost any type of application or backend service, and everything required to run and scale your application with high availability is handled for you.

Serverless enables you to build modern applications with increased handiness and a lower total cost of ownership. Building serverless applications means that your developers can focus on their core product instead of worrying about managing and operating servers or runtimes, either in the cloud or on-premises. This reduced overhead lets developers reclaim time and energy that

can be spent on developing great products which scale and that are reliable. According to NIST, PaaS is defined as

> the capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider. Here the consumer does not manage or has no control on the underlying cloud infrastructure which includes network, servers, operating systems, or storage, but rather has control over the deployed applications and any other possible application hosting environment configurations.

With this definition, users are expected to manage the distribution of applications and have control over accommodating environment configurations. With Serverless FaaS, user control over hosting is removed. This paves the way for simpler scaling and a more attractive billing model. Here the cloud provider has control over the hosting environment's configuration and runs user-provided code only when it is invoked, and only. That is a significant change when compared to that of PaaS. FaaS is very attractive for PaaS users as they do not need to pay for idle resources and avoid managing rules when it comes to auto-scaling. The main thing that distinguishes serverless platforms is transparent autoscaling and the process where you are charged only when code is running.

Serverless computing is rapidly gaining attention from IT practitioners and academics alike. Serverless computing is an emerging cloud computing paradigm that provides a platform to efficiently develop and deploy applications to the market without having to manage any underlying infrastructure. Everything shows that serverless computing is the future and the way

to go, there is still more to develop and improve upon this new and evolving technology with regards to its enormous benefits and potential to change the face of information technology.

Summary

This chapter briefly addresses serverless technology security and privacy issues. Background related to the problem was discussed which addresses the latest and past security incidents that have taken place in serverless technologies. Later the literature review was discussed which talks about the other research paper that has put focus on serverless security features. It was divided into two parts i.e., literature review related to the background and literature review related to methodology. The literature review related to the background discusses everything on what is serverless computing, the types, and characteristics of serverless technology, where this technology is being used, and how it is used. The literature review related to the problem discusses the security problems of serverless computing, the incidents that have taken place, and discuss the security and privacy issues that are studied by other researchers.

Chapter III: Methodology

Introduction

The current chapter discusses the plan to address the issues that were mentioned in the previous chapter. I did investigation to achieve desired goals. Some of the questions that are addressed has been mentioned below

1. How to detect and identify vulnerabilities in serverless computing?
2. Different problems are mentioned in the previous chapters. How can we drive solutions to the mentioned problems?
3. There are different tools available such as Open-source tools like SecLambda. What are the ways to protect those tools?
4. How to protect the systems that use serverless technologies?
5. How should users or developers take care while using the serverless technology?
6. What are the Security and privacy issues concerning serverless technology?

The above questions provide a pathway for the data to be collected, tools and techniques to be used, and help to analyze and interpret the data.

Design of the Study

Q1 addresses identifying vulnerabilities in serverless technology and how to overcome those vulnerabilities and protect the systems. This is found in Google scholar, news articles, and blogs from the time of 2015 till now. This study helped me to achieve the bugs in the technology that causes data losses or other security issues. Once, the vulnerabilities are identified and resolved the serverless technology becomes a useful tool for the users or developers to use. The keywords that helped me to identify the expected results are Vulnerabilities in serverless,

security and privacy issues in serverless, serverless latest attacks, what are the measures taken by the serverless providers to overcome the vulnerabilities in their systems.

Q2 was approached by reading the problems that are mentioned in chapter 1, understanding the problems, and analyzing the solutions. This was achieved by doing a thorough investigation in google scholar and news articles. News articles and blogs related to serverless computing discusses the open problems that are encountered in serverless. This is significant as the problems are the core of this research paper and aid to identify the solutions when the issues are recognized. The keywords that helped to answer this question are Serverless security and privacy issues, how to overcome security issues in serverless technology, latest incidents in serverless computing. (Latest includes incidents that have taken place from 2015 till 2021.)

Q3 was addressed by conducting a thorough investigation using the article provided by the researcher who has introduced the SecLambda Tool. The keywords that helped me to identify the potentialities to protect this tool are SecLambda tool protection, SecLambda security issues, design methods. Google Scholar helped me to find the paper that presents this tool. The timespan was 2020 as this tool was introduced recently

Q4 discussed on how to protect the systems that use serverless technologies. This was done by examining the system's that uses serverless, the policies that those systems or organization supports and the implementation methods of serverless computing. By analyzing the data, the vulnerabilities were identified, and solutions or improvements rendered the systems. The keywords that helped to solve this question are a system that uses serverless technology, serverless technology service providers, serverless architecture, Using serverless technology to a

platform. The timeline for the search was most advanced, i.e., 2019, 2020, and 2021 as the system and protection against the systems keeps evolving.

Q5 discussed how users or developers should take care while using serverless technology. This was investigated by reading the serverless provider's websites as they talk about the security and privacy issues of their tools. This acts as a knowledge input for the developers to use the serverless system considering those facts provided in the serverless provider websites. This tender's developers use the application without any vulnerabilities. The timespan of this search was from 2019 as the latest results yield the most advanced outcomes.

Q6 talked about the Security and privacy issues concerning serverless technology. The results assisted the reader to understand the concerns in serverless technology, thus further serves to settle the issues based upon the yields attained in this study. The developer can build the application using serverless by acknowledging these concerns. The time span of the study started from the period where serverless came into existence, i.e., from 2015 as we discussed the past issues, latest issues and analyzed them. The results were found in google scholar, blogs, news articles, serverless provider websites like AWS lambda which has documentation on the security issues that were being addressed.

Data Collection

As mentioned in the design of the study, data was collected using Google Scholars, Blogs related to serverless, new articles on serverless incidents, serverless provider websites as they discussed more the security and privacy features of their tools, journals, etc. The data was listed as a taxonomy that contains serverless security and privacy issues and countermeasures. The period started from 2015 as serverless first introduced in 2015 by amazon which is named AWS

Lambda. Firstly, I investigated the questions that were discussed in the design of the study, later I analyzed the collected information and classified it in a taxonomy. This process will be explained better using the following diagram.

Figure 2

Data Collection

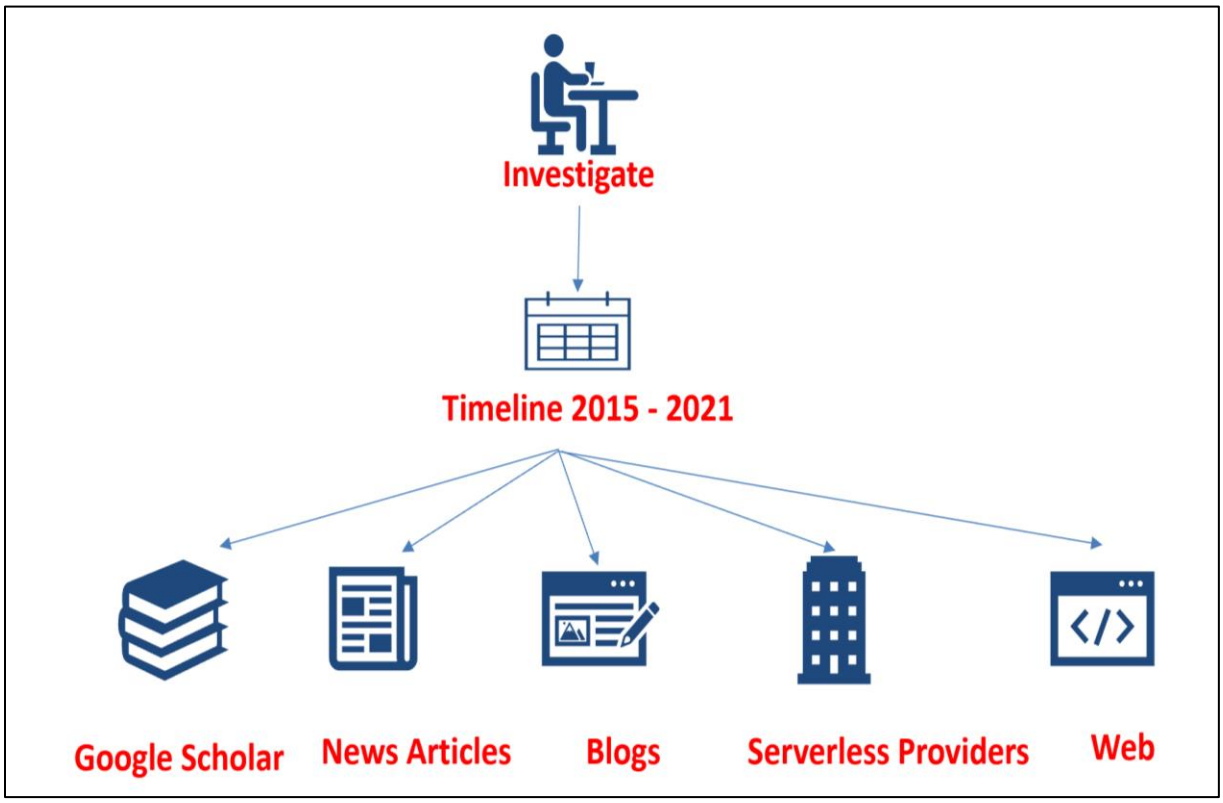


Figure 3

Analyzing and Interpreting the Data

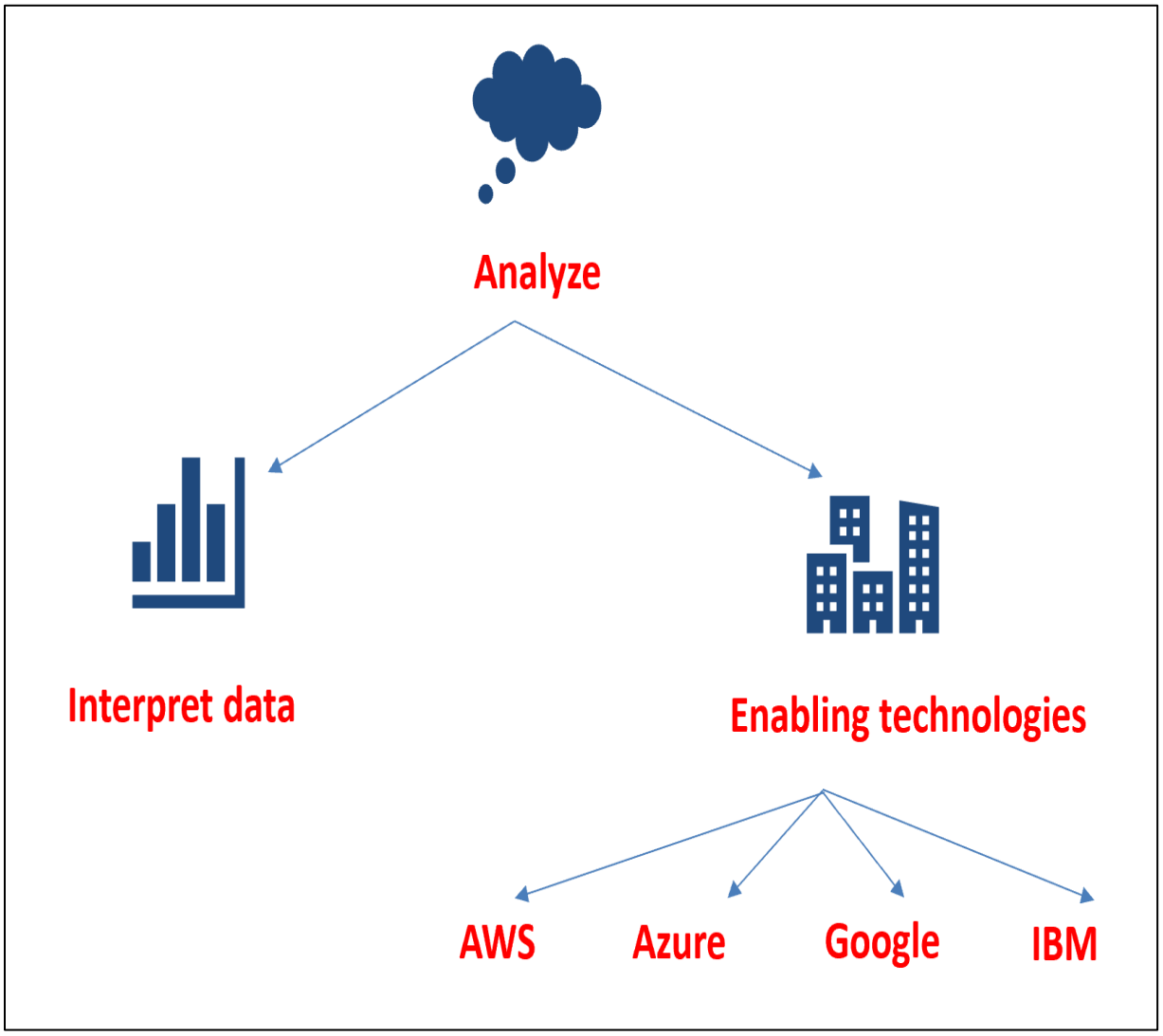
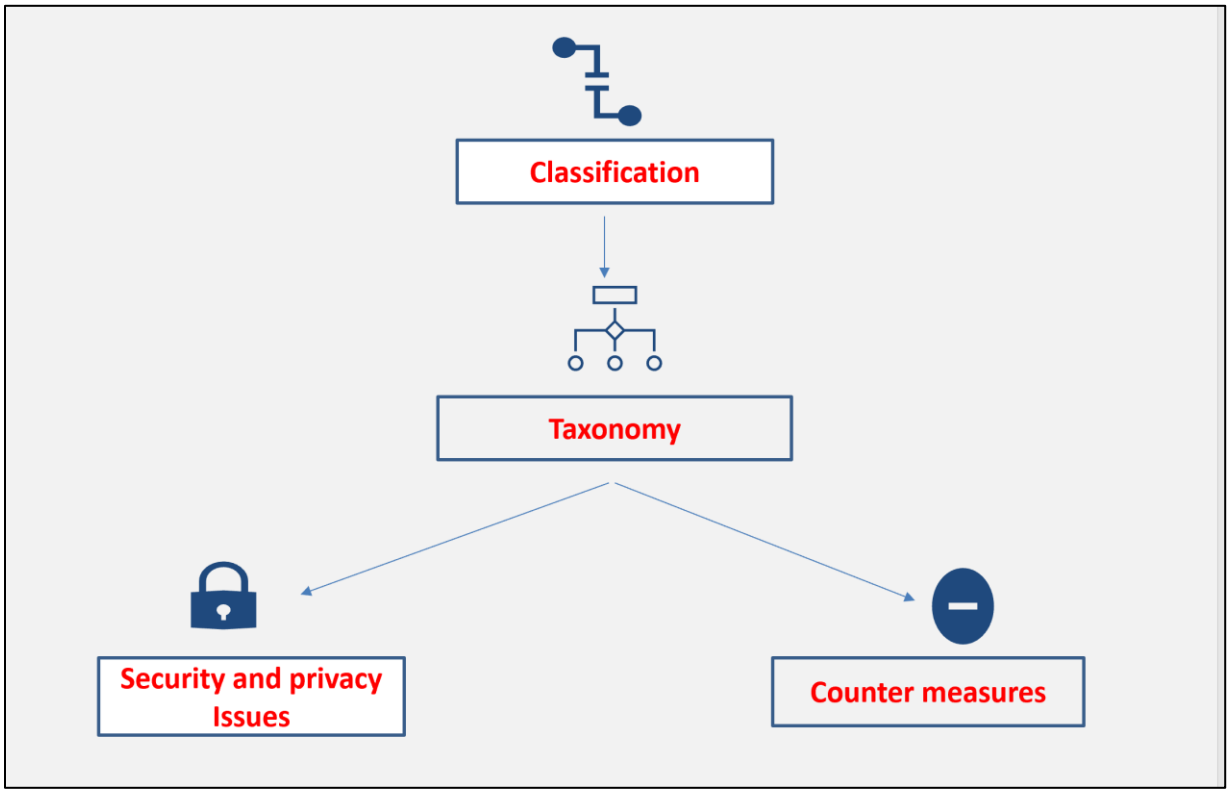


Figure 4

Classification of the Data



Chapter IV: Data Presentation and Analysis

Introduction

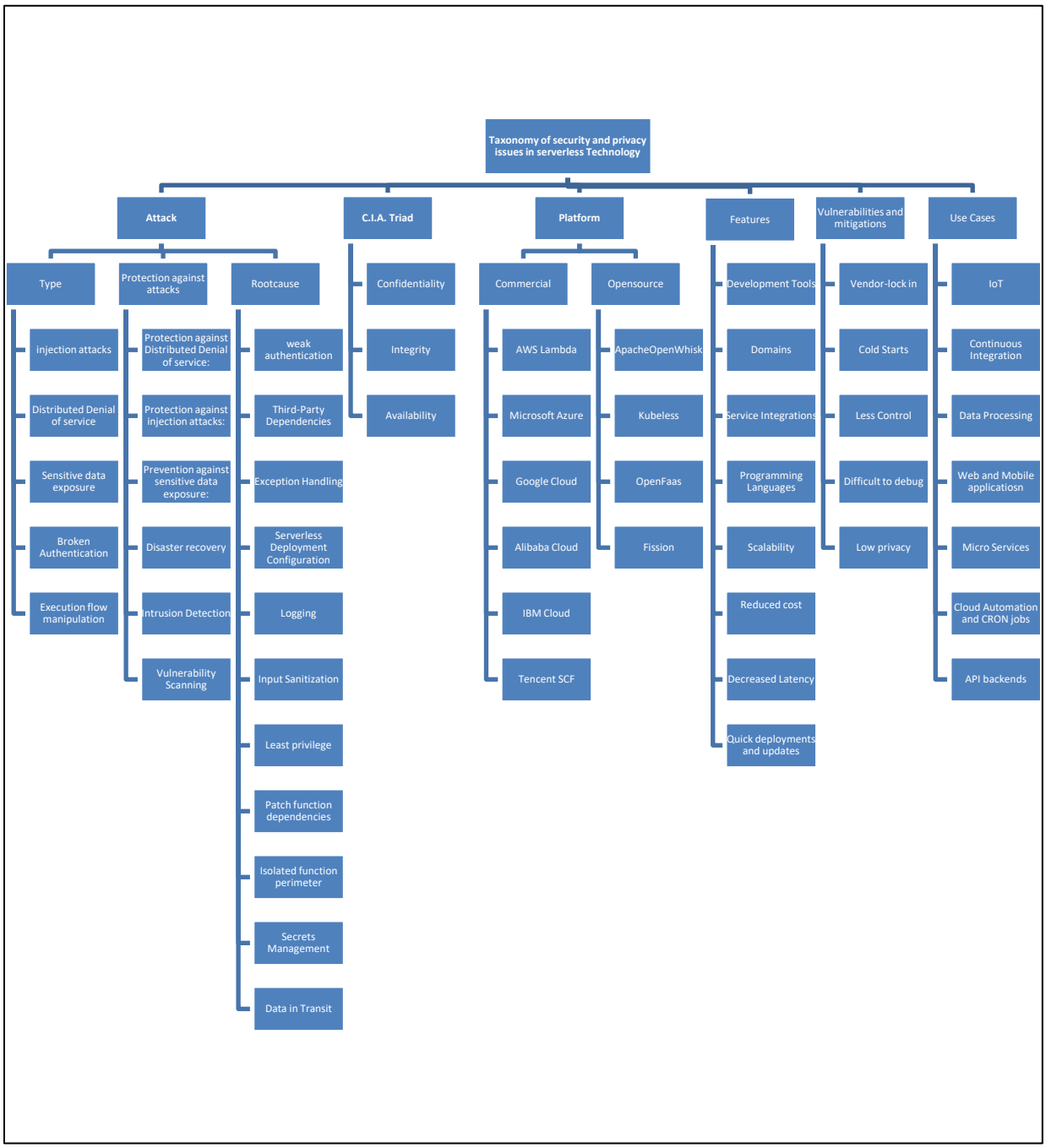
This chapter clearly present the in-depth analysis of the serverless technologies, the taxonomy that defines its security issues, providers, use cases and mitigations to the security issues in serverless computing. Based on the design each attribute is defined and elaborated.

Data Presentation

The taxonomy below contains several branches, and each branch is divided into sub sections. Each section clearly defines the security and privacy issues in serverless technology, what are the root causes of those issues, the platforms that the technology is built upon, its features and use cases. Figure 5 explains the overall features of serverless, and Figure 6 describes the Development tools.

Figure 5

Taxonomy of Serverless Computing



The above picture depicts the serverless computing taxonomy that consists of different attributes. As shown in the picture it is comprised of several dimensions such as Attack, C.I.A. Triad, serverless platforms. Each dimension has been divided into several subcategories. The C.I.A. Triad has been defined as Confidentiality, Integrity, and Availability. The attack is divided into Attack types, Protection against the given attacks, Root cause, and the attack impacts. The platform is then categorized into commercial and Opensource platforms. Subsections are explained below.

A. Attack

This paper talks about the attacks that occur regularly in serverless computing.

i) Attack Types:

1. Injection Attacks: The attack that allows users to enter malicious inputs that alters the program execution. This happens when the inputs are performed without proper verification. Lambdas run on operating systems and send HTTP requests to the URLs. Similarly, the attacker can perform the same and send the variables to some random link. This leaks sensitive data (Gillas, 2019). The event data which are the additional inputs to the serverless application cause possible injection attacks. There are several types of injection attacks that can occur in serverless technologies.
    a) Event Data Injection: The serverless technologies perform in a way that it accepts even inputs. This in turn increases the possibility of data injection attacks. HTTPS or API calls can be hijacked. The security tools are not yet acclimated to this kind of vulnerability testing.

b) Command injections: This injection occurs when the commands execute against the operating systems. The attacker passes the commands through the URL and reads the contents of the file causing injection attacks.

c) SQL Injection: SQL injection is a very common injection attack across the web. This occurs when the hacker obtains the records in the database bypassing SQL queries in the input filed. The attackers can delete, modify, and can execute admin operations. The event sources increase the attack surface.

d) Cross-Site Scripting (XSS): This occurs when the inputs given by the user are not being validated before accepting. JavaScript has a higher chance of XSS including java, ActiveX, and VB Scripting (Yagolnitze, 2020). Emails and cloud storage can lead to this kind of attack in serverless technologies.

2. Distributed Denial of service: A Denial-of-service attack is to make the resource, website, or application unavailable. DDOS also makes the website unavailable by flooding with the traffic using multiple sources. Default limits or executions make the serverless vulnerable to DDoS Attacks. Defaults such as pre-memory allocation functions execution limits, number of pre-defined processors, and disk capacity (*Security risks arising from serverless technology*, 2021). HTTP flood attacks are the most common DDoS attacks where a huge amount of HTTP requests occur. The network is infected by bots and the group of bots is referred to as zombies. The botnet sends requests

to the IP address and interrupts the normal traffic flow which in turn results in Distributed Denial of Service attack. Denial of the wallet is quite similar to the DoS attack, but this attack leads to the financial loss of an individual since it targets mostly serverless users. This Dow will make the site available but leads to bankruptcy. If the vendor serverless is insecure, this leads to the Dow attacks.

3. Sensitive data exposure: It is defined as thieving legible data, stealing keys, or executing attacks such as man-in-the-middle attacks. The attackers can read the database tables, access cloud storage such as s3. The most common cause for sensitive data exposure can be not using encryption techniques such as cryptography protocols, storing the data in the temp directory, and not deleting the data after use can lead to weakness.
4. Broken Authentication: Attackers can bypass the logic of applications and manipulate their flow when there is weak authentication. It also exposes potentially executing functions as well as performing actions that were not supposed to be exposed to unauthenticated users. Developers are advised not to commit resources in building authentication systems but rather use authentication systems provided by the serverless vendor. Broken authentication occurs when the design is poor or when the access controls are not in place. Attackers use the resources such as cloud, open APIs that can lead to exploiting the internal resources, data leakage, and break the flow execution.

5. Execution flow manipulation: The hackers subvert the logic which in turn modifies the application flow. This leads to denial-of-service attacks and access control bypass. Since the serverless technologies follow the microservice design patterns and this follows a certain flow of execution. If a malicious user enters the system at a certain step, he will consume all the available resources of the system and deny other system user services. This can be prevented by having access controls in place rather than assuming the execution flow (*The ten most critical risks for serverless applications v1.0*, 2018/2022).

ii) Protection against the attack:

1. Protection against Distributed Denial of service:

   A Denial-of-service attack is to make the resource, website, or application unavailable. DDOS also makes the website unavailable by flooding with the traffic using multiple sources. This can be prevented by following measures such as configuring WAF rules such as firewalls, using the AWS shields, blacklisting the traffic that comes from specified locations, and configuring the query strings to clock certain kinds of requests, protecting all the endpoints (VISH, 2020). Sometimes issues occur from the single IP address or traffic spikes at some odd hours. Those IP addresses can be blocked.

2. Protection against injection attacks:

   This attack permits the users to document malicious inputs that alter the program implementation. This can be prevented by following various

measures. Updating libraries for the programming languages such as java and .net to the latest version decreases the occurrences of injection attacks. Validate all the user inputs before accepting them. Limiting the privileges and user access. implement access restriction, by following least privilege principles. Encoding all the data before the client receives it and using valid headers can prevent XSS attacks. Validate all the special characters as they can lead to SQL injection attacks. The utilization of Web Application firewalls (WAF) allows scanning all the incoming traffic for the serverless applications. This works by performing behavioral analysis to detect intrusions and hacking, threat blacklisting, and application whitelisting (*Web Application Firewall (WAF) solutions*, 2022).

3. Prevention against sensitive data exposure:

    The serverless technology can be prone to data exposure. This can be protected by classifying the sensitive data and rescuing data at rest. Reserving sensitive data should be minimized. Utilizing the HTTP endpoints for the APIs can render in preventing sensitive data exposure. During function runtimes, using environmental variables provided by the infrastructure providers prevents data exposure (*owasp-top-10-serverless-interpretation-en.pdf*, 2017). AWS introduced a tokenizer solution that helps to prevent data exposure. Token is different from encryption. In encryption, an algorithm is used to change the plain text to ciphertext, and keys are used to decrypt the data. Here in the tokenizer method, the data is transformed into a string of

characters called tokens. These act as a reference to the original data (Beswick, 2020*)*.

4. Disaster recovery:

Disaster Recovery helps to deprecate the loss that is transpired due to natural disasters or catastrophic events. Serverless computing has a backup plan in such critical circumstances. Disaster recovery means minimizing the impact of disasters by having a proper backup plan that limits the downtime and plans to minimize the data loss during the events of disaster. From the Engineering perspective and risk management, Disaster recovery plan two falls into place.

1) Recovery Time Objective (RTO): Usually every company or service maintains 2nd production service to use during disasters or primary production downtime. RTO has calculated such that the length it would take to move to an alternative production service

2) Recovery Point Objective (RPO): this is the portion of information loss that is calculated in a period (Tseggai, 2018).

Serverless companies such as AWS lambda have automated backups, documentation that help during the disaster periods, automated deployment tools, and automated and continuous monitoring which limits the impact during the disaster (Allen, 2022).

Usually, the CEO and the related level people of the company execute the Disaster recovery plan by calculating the two things. Once the goals have

been set and the process has been defined, they initiate the Disaster recovery plans. The recovery steps include Storing the data and restoring the data from the production which is being impacted to the safe production environment. They restore all the cache, data, authentication, authorizations. The backed microservices will also be restored, API endpoints will be swapped, and verify that the recovery is complete. With the Disaster recovery, the companies ensure continuous improvements from the current, past actions to ensure that the product is stable in the future.

5. Intrusion Detection: Intrusion can be detected and prevented by using traditional firewalls. The serverless doesn't use a physical server, which makes them unable to use the traditional firewall or IDS/IPS. Serverless used WAF (Web application firewall) to protect the applications against common exploits. The WAF contains IP addresses lists that need to be blocked. It also contains a common attack pattern that blocks common attacks. NACL's (Network ACL) can be used to define rules, which are more powerful than WAF and controls traffic. One NACL can have 20 rules. It operates as a firewall. If a user needs to block a certain IP address in a given range, then NaCl could be the better option (Budzoń, 2017). Whitelisting can also help in authorization certain IP addresses.

    AWS introduces Guard Duty which is a hazard detection service that scans the system for malevolent activities. This analyzes and monitors API calls. This detected attacks at the network level as the traditional IDS systems

do. This uses a machine learning technique to detect network anomalies and prioritize the potential threats. Once this finds certain anomalies, it stops unauthorized activity (*Intelligent threat detection—Amazon guard duty*, 2022).

6. Vulnerability Scanning: There are various tools that are used to scan the vulnerabilities in serverless computing. Some basic API tests help to run tests on the servers (Perris, 2021).
    i. DAST: Dynamic Application Security Testing runs all the predefined inputs and checks the vulnerable pieces of evidence. The response is measured based upon the exploits.
    ii. The software Composition analysis will check the dependencies with the known security vulnerabilities. Sometimes the third-party dependencies will bring more vulnerabilities than the actual application. The software composition analysis will detect such types of vulnerabilities. the NPM audit is one such helpful tool to detect third-party dependency vulnerabilities.
    iii. Synk: This is one prevalent tool in serverless that scans for vulnerabilities and restores them automatically. Many prominent organizations utilize Synk to produce secured applications. It integrates easily, scans continuously, and provides quick-fixes (*Developer security. Develop fast. Stay secure*, 2022)

iv. Static Application Security Test (SAST): Sometimes vulnerabilities can happen with the source code. This occurs due to wrong coding practices. Tools such as GREP, ESLint, Sonar Lint can detect such kinds of vulnerabilities.

v. Aqua: This tool easily integrates with the CI/CD pipelines and protects AWS lambda functions in the runtime execution environment (*Serverless Security for serverless containers and functions*, 2022).

iii) Root causes:

1. Weak Authentication:

Attackers can bypass the logic of applications and manipulate their flow when there is weak authentication. It also exposes potentially executing functions as well as performing actions that were not supposed to be exposed to unauthenticated users. Developers are advised not to commit resources in building authentication systems but rather use authentication systems provided by the serverless vendor eg. AWS Cognito or single sign-on (SSO), AWS API Gateway authorization facilities, Azure App Service Authentication / Authorization, IBM Bluemix App ID, or SSO.

2. Third-Party Dependencies:

Serverless technologies use third-party software or API and libraries. Some of these might already be prone to security attacks or might be vulnerable. This can in turn lead to security issues in serverless technology.

The only way to protect from this kind of attack is to regularly update the libraries/software/APIs that we use from 3rd parties.

The incident that took place in the capital one in August 2017th (Capital One reports inside job data breach, 2017), which was arisen due to employing the 3rd party libraries that are not updated. The incident resulted in the exposure of sensitive data information like Date of birth, personal identification number and compromised several files that contain customers' data.

Hence, even if the system that we use is highly secured and if the dependencies that we have are vulnerable, security issues tend to occur. Hence to prevent such kinds of security issues, the 3rd party dependencies should be used only if necessary and need to be updated regularly. The software/libraries need to be scanned on regular basis to see if they are any security breaches or changes in the workflow (Rehemägi, 2021a, b, c)

3. Exception Handling:

Debugging is complex in serverless and as such developers rely on and adopt verbose error messages in the form of syntax errors and stack traces and variables which enable debugging environments. Problems occur when in the process of moving into production stage codes are not cleared, this reveals details about the serverless function internal logic which mistakenly exposes sensitive data. In other to avoid this, developers are encouraged to avoid

verbose messaging but rather rely on debugging systems provided by the vendor's architecture.

4. Serverless Deployment Configuration:

   With serverless and its architecture being new, there is a high chance of misconfiguration whiles setting up environments is high and can lead to data loss as well as exposure of sensitive data.

5. Logging:

   A peculiarity of serverless computing is that functions and containers are turned off and on in a matter of seconds. If the state, status, and other information related to these functions and containers are not captured, the information is irretrievably lost. In the event that a block of code written in a temporary container malfunction, how can DevOps verify that the right events required to trigger the block of code were implemented for example? This is where the importance of logging becomes paramount. To understand how an application behaves, what happens when it fails, logging is compulsory.

6. Input Sanitization:

   The Open Web Application Security Project (OWASP) ranks injections as one of the most dangerous security risks in both conventional web applications and serverless applications. Serverless environments by nature have various input sources. For example, AWS lets users/developers use multiple input sources to call their Lambda functions. These inputs can be from databases, API gateways, file storage, etc. Users/developers must imbibe

the use of input sanitization best practices both when writing the program and when it is in operation.

7. Least privilege:

   privilege in serverless computing/development is a lot more granular than other cloud systems. Users/developers are able to set permission models on smaller blocks of code (or function). When deploying many functions, the easy method is to default to the maximum permission levels allowable for each function. This practice leads to functions with less-than-optimal permission security. Users/developers should practice the principle of 'least privilege' when managing permissions giving each function the minimum permissions necessary.

8. Patch function dependencies:

   In serverless development, Function as a Service (FaaS) environments have the responsibility of patching operating system dependencies in your environment. However, the responsibility of patching your application dependencies lies with you, the developer. The application dependencies and third-party code libraries pulled and used for software development such as npm, PyPI, Maven, etc, all are examples of dependencies whose patching responsibility lies with the user. A major example was in the case of the Equifax Credit breach, a known bug in a dependent library was exploited to gain access to their systems.

9. Isolated function perimeter: As we know, serverless computing uses function as the unit of scale and developers simply develop their pieces of code then submit them to the cloud provider for their execution. The serverless environment the cloud provider provides is secured using several security mechanisms and techniques. The developer's functions are the inputs to this secured environment and the cloud provider is not responsible or accountable for the security of those functions. In that case, if the developer uploaded a function with security vulnerabilities and when it gets compromised that function can put not only the developer's data in the function at risk but also the other developers' functions and cloud environment resources. One mode of mitigating this risk is maintaining isolated function boundaries. Sometimes the deployment of more than one function is needed to create the workflow and get the expected output. But we cannot assure that every function in the cloud environment is highly secured. For instance, a function that sanitizes its input from the users which neglects input sanitation from another function can get compromised due to the other function not sanitizing its inputs from the user and getting compromised already. Isolated function perimeter assures the functions are isolated within their perimeters (Tal & Podjarmy, 2019a) and treats every input to the function coming from a suspicious data source. This way it can filter all the inputs and block harmful data coming into the function and reduce the risk by isolating the compromised function from harming other

functions. Creating and adopting standard security libraries also helps ensure isolated function perimeter.

10. Secrets Management:

    Since serverless functions are just pieces of code most of the time they need to get access to other resources from any other cloud provider. In that case, serverless functions use credentials (Naor, 2018). Depending on the type of service such as cross-account integration, these secrets can be short-term or long-term and must be saved at someplace. This is where we require an appropriate method to access, and control secretes. From the developer's side, they can focus on encrypting their secrets and store them in a separate secure place in the memory and scan the code regularly for any accidental commits of secrets. From the serverless provider's side, they offer integrated tools to fulfill this requirement which is called secrets management for added security. This is secure storage where the developers can store their secrets and sensitive credentials. For example, AWS Lambda employs the "Secrets Manager" whereas Azure utilizes the "Key Vault". This helps in reducing the risk of sensitive information exposure to a great extent. During the deployment of the functions, serverless applications can call the API of the secrets manager and consume the secrets they need such as access keys. Using secure storage for sensitive information is highly encouraged and recommended in serverless environments instead of saving secrets in static files in code repositories and environment variables. Bitcoin mining incidents

that happen in 2014 (Mardesich, 2014) and 2015 (Pauli, 2015) are good examples for using secure storage and maintaining good security practices. Moreover, Keeping the secrets in secure storage is the serverless providers' responsibility and they need to make sure that the data is encrypted.

11. Data in Transit:

    Since we are articulating about "serverless", individuals may believe that there is no data, and it is just a function that is supposed to do some tasks. But there is data within functions that are fed by the developer using variables. This data can be user credentials, access keys, credit card information, and any other sensitive information. We do not recommend adding sensitive information into functions but, developers can make mistakes. However, the risks arise whenever the data is transmitted and exchanged. This transmission can happen in many ways. One is when the developer uploads the coding to the serverless environment. Another way is during the function's execution. Depending on what the function's task is, it may need to connect to other functions in the serverless environment or any other third-party vendor to get access to some other services(*Many-faced threats to serverless security*, 2017). Whenever the data is in transit, it increases the attack surface. This risk can be mitigated by making sure the medium that is used for web communication is sufficiently secured. Serverless providers force developers to use secure protocols such as HTTPS whenever the developer makes a connection with the serverless provider to submit their pieces of code.

Moreover, using SSL certificates to verify the identity of whom they are communicating with is also highly encouraged in serverless computing (Tal & Podijarny, 2019b).

B. **C.I.A. Traid:**

Confidentiality, Integrity, and Availability ate the three main pillars of information security. It is always a best practice to have this CIA Triad in hand for all the technology implementations. Serverless can be secured by following these most promising approaches.

i) Confidentiality:

Confidentiality means protecting sensitive data. This can be achieved in several ways such as data encryption, access control, File permissions, etc. This is executed by restricting unnecessary access for the sources. Serverless functions should be imposed with the least access privileges. Separation of Duties should be considered such as separation of policies and separation of roles. This can help in exposing sensitive data and unauthorized access. Only acceptable services should have interacted with the serverless functions. If all the services were granted access, the vulnerabilities in those services could make the serverless system weak even if the function is more secured. Network inbound and outbound should be restricted for selected sources and destinations (Khani, 2021).

ii) Integrity:

Integrity is defined as unaltered data, i.e., protecting data from transformations and omissions by unauthorized parties. Because if the data is changed, the damage

can be irreversible considering the sensitivity of the data. Serverless deals with data processing, IoT, etc. which have a lot of sensitive data that it deals with. What if the data is being altered which in turn causes severe problems to the users such as incorrect medication prescription or devise malfunctioning which could take out the life of an individual? Therefore, integrity is one of the primary characteristics of the CIA Triad (What is the CIA triad? 2021).

This can be achieved in serverless by following best practices such as continuous monitoring of the data and observing if there are any sudden changes. Logging of all the actions is another best strategy. The function data must be encrypted even if data in transit or at rest. This way data alterations will not be attainable by unauthorized people. Regular audits, log files, and taking benefit of all the monitoring tools supplied by the serverless providers helps in having data visibility that in turn achieves integrity.

iii) Availability:

Availability is defined as the data being available for actual use whenever needed. An example of lack of availability is a denial of service (DoS attack) where the service is down by sending multiple requests at a time and making the service unavailable. This can cause huge financial loss for the business. Hence ensuring Availability is an essential task.

Imposing limitations on computation memory, concurrent executions can aid in controlling runaway functions which in turn cause DoS attacks in serverless. The serverless providers such as IBM have Disaster Recovery plans in high if

there are availability issues. Hence there can be very low downtown in case of any disasters. availability can be achieved by taking a few precautions such as risk analysis, dependencies documentation such as Recovery Time Objective (RTO), and Recovery Time period(RTP), Backing up Data, etc. (*disaster-recovery-introduction*, 2019)

AWS Lambda has the built-in highly available feature which makes it's one of the best serverless provider to consider (*Serverless computing–Amazon web services*, 2022).

C. Platform

i) Commercial

1. AWS lambda:

This service was provided by Amazon. This is an event-driven, serverless platform. Using **AWS Lambda** developers can run their code virtually without any administration. Developers can write their code in the supported languages, create functions, applications or can upload their own code. The lambda functions can perform different computational tasks. Lambda will automatically manage the resources with high availability, scalability, and security. Also, developers can use the other AWS service with the same login to extend the services. It supports most of the programming languages like Java, Python, C# (. NET Core), and Node. Js. This service can be found in the AWS website linked here (*AWS Lambda – Serverless compute - Amazon web services*, 2021)

The AWS lambda uses certain criteria to work such as functions, events, services, and resources. The function is something like microservices and processes the files, scheduling the tasks, and maintaining the user's data in the database. The event must be triggered to invoke a function. This can be a user request, message, or image. Once the function is invoked, the resources are being used such as AWS s3 bucket, Dynamic DB table, etc. A service is a project that the user can utilize to build the applications after a function is invoked by the event and the resources being allocated. The service can be in JSON format or the YAML format. (Serverless framework–AWS lambda guide–Introduction, 2021).

2. Microsoft azure:

This service was developed by Microsoft which makes their users build applications without worrying about the infrastructure and administration. *Azure functions* help to select different programming languages based upon the requirements. The programming languages are selected depending on the runtime. C#, JavaScript, and F# are supported by 1. x runtime while C#, Java V8, and Python 3.6 are supported by 2. x runtime The APIs used b azure are developer-friendly enabling the developers to build their code and deploy faster. Azure also concentrates on team performance, i.e. it helps to boost the team performance by providing a fully managed platform so that team can build their applications. Azure is an event-driven serverless platform that

helps to solve complex problems. The commercial page can be found here (Azure serverless. Microsoft azure, 2021).

Azure can be used to test the application or the functionality or the business logic (Serverless framework–Azure functions guide–Testing 2021). It creates endless connections and boosts productivity. Users can use Azure Cosmos DB to utilize the database. Workflow can be made easier using Azure serverless. This uses API management and creates seamless workflow (Gursimran, 2020). The main approaches that azure serverless follows are Azure logic apps and Azure functions. The logic apps help to create workflows and the Azure functions to build the applications (Chris, 2021)

3. Google cloud:

The Google cloud functions is a serverless technology developed by Google. This helps to write simple single-purpose functions, run codes locally or in the cloud without worrying about the infrastructure. This is an event-driven serverless compute platform. This helps from coding to deployments. Like the above serverless services, it also enables users to pay as they use. Google cloud functions manage code as well as infrastructure. *Knative* which is managed by Google Cloud an open API, helps developers to run their workloads from anywhere. This serverless service supports multiple languages like Java, C#, Python, Node.js. Knative helps the developers to focus only on the coding part, while it takes care of all the deploying, building, and managing services (Google Cloud, 2021). This tool supports

various frameworks such as Spring, Ruby on Rails, and patterns such as GitOps. This provides control by making the technology built with the Continuous Integration/Continuous Deployment (CI/CD) which can run anywhere whether it on servers or cloud.

The Knative routes all the traffic that occurs during the deployments and scales automatically. Abstraction is achieved which makes the code reusable. Knative can be integrated with the user's own platform while using all the benefits of the Google serverless technologies (Manor, 2018). The Knative reduces vendor-locking's. The google cloud is integrated with other services like artifact registry to manage images and packages and Docker hubs. The main advantage of the serverless is parallel computation i.e., when there are many batch jobs that need to run and are intense, the serverless schedules those jobs (*Bertram*, 2021). In the GCP, the user sends an HTTP request to the google cloud function, the google cloud function sends the response or the URL immediately, but whereas AWS lambda uses API gateways in between the communication as a medium. The logic or the code is wrapped in a function that makes it executable in the GC function. Each of the modules performs single action which is quite similar to microservices architecture (Aruljothi, 2021).

4. Alibaba Cloud:

Container technology helped developers build and deploy their functions, but the disadvantage is that it might take a very long time to scale out the

instances. Hence there is no auto-scaling management. Hence, serverless came into existence. Alibaba provides the tools for the developers to implement serverless features. "*Function Compute"* is one such serverless technology created by the Alibaba cloud. This ensures low operational costs but delivers quality business deliverables. This is event-driven and can be used to build applications using the services. The function compute runs the code elastically and in a reliable manner(*what is serverless computing? what are the features of serverless?,* 2021)

The Official serverless technology of Alibaba cloud which is known as Function compute can be found here (*Function compute–Alibaba Cloud*, 2021a, b). This service contains several features such as an event bridge trigger in which it routes all the events to function compute. HTTP trigger is another event that recieves and processes HTTP requests and sends back the HTTP responses. This supports various programming languages such as Java, Python, Node.js, PHP, c#, etc. To perform deployments, debugging, and run the application various development tools are available such as funcraft which is a tool that helps to debug local, create the resources, and utilize third-party resources. Similarly, fcli is another development tool which is a command-line interface that is provided by Alibaba cloud to manage the resources that are in Function compute. Visual studio code extensions also help to create, debug, or deploy the applications. Several instances and resource types are available in Alibaba function to compute that helps to increase the computing

performances. The billing is charged based upon the amount of time the resource is being used and is not charged for the idle time which makes the serverless cost-effective.

The workflow in the function compute is as follows. The developer chooses the programming language and compiles the applications. Then he uploads the application in the Function compute. This contains either SDK or API. Then the trigger calls the function. The trigger method contains functions to compute, APIs, logs, table stores (*What is function compute*, 2018). The function computes resize dynamically based on the request and the billing is invoked. The main advantage of Function compute compared to other serverless providers is that it uses event triggers, log queries, monitoring, and alarms for troubleshooting. scaling is done within milliseconds to an accuracy of 100 milliseconds. As per Wang et al. (2021), the Function as a service application use container tools and images for deployments and invocation. Cold starts occur in serverless computing as it takes time to fetch the data or the images in the containers. The cold starts range from seconds and might take minutes to start.

5. IBM cloud:

It is developed by IBM based on Apache Open Whisk. This is a FaaS functions-as-a-service (FaaS) programming platform. This enables developers to develop lightweight code and build applications. Here users can pay for what they used. It supports many programming languages like Python, Swift,

Node.js, PHP, and Java. The IBM cloud functions provide greater control and scalability as IBM has a network of networks that segregate the traffic and streamlines. The international network has 2,000Gbps of connectivity which in favor operates independent network streamlining administration (IBM Cloud. 2021).

The serverless applications are deployed in containers. The use cases that support serverless are Data Processing, Parallel computing, API backend, stream processing workloads, and microservices. Data processing states that the serverless is quite adapted with the audio, texts, images, data, and videos. Parallel tasks can be run at one invocation like data search or processing which makes the serverless parallel compute. API gateway provides extra security. this enables the HTTP to be readily utilized by the web clients and available for web actions (IBM cloud education, 2021a).

Web applications, mobile applications can be built using this service. The user accesses the application in the object storage. The API is called when the application is used which is defined in the API gateway. This sends the request to the cloud functions. functions can be developed using various languages. The tasks can be scheduled in which the function can be executed periodically. The mobile developers can access the server-side logic and implement the functions in the languages that like swift to consume server-side functions. The commercial IBM cloud functions can be found in this website (*IBM Cloud Functions–Overview*, 2021)

6. Tencent SCF:

Tencent SCF is a Serverless Cloud Function that helps to develop and deploy code. This is an event-driven, serverless platform. It provides auto-scaling and helps the developer to build the application and supports most of the phased-in software development lifecycles such as coding, deploying, debugging, and testing. It also extended its service to alarming, monitoring, and troubleshooting. Once the resource is available, the developer can simply write the code.

SCF reduces overhead time as the billing is done based upon the number of requests and the resources being used. It has a centralized architecture that makes deployments and tests done in just one click. The code can be written without worrying about the components, which makes the SCF easy to use. Even though the volume is high, SCF provides resources to meet the client's business needs. The SCF is compatible with various frameworks. Once the function is written, it can be deployed automatically which makes SCF high in efficiency. SCF has resources in various locations, making sure that if there is downtime in one resource due to any reason, it automatically provides the resources from another location. This makes the service reliable. The official website to use the Tencent SCF can be found here (*Serverless cloud function. Tencent cloud*, 2021).

SCF has high fault tolerance. In order to run functions, users can customize the time to run those functions. This uses Ckafka messaging

system, SDK is used when an application is being called. It supports various programming languages such as Go, PHP, Java, Python, Node.js, etc. The functions can be written online or offline using the web IDE's that are supported by SCF which makes it easier and more convenient for the developers to use. It allows developers to connect to git. Tencent uses push and pulls trigger modes. The trigger pushes the events and pulls events from function execution. The trigger events are represented in JSON format. The input parameters in the programming languages such as java contain objects that matches the event (Cloud, 1998)

ii) Open Source

1. Apache open whisk:

    This is an open-source serverless platform. It executes functions in response to the events. It helps to perform multiple operations in the code. This allows writing functions in the language desired by the developers like Java, Scala, .NET, PHP, Python, Ruby, Rust, NodeJS, Swift and Go. The Apache open whisk provides developers to integrate with other services. This provides scaling and optimal utilization. It also provides a development tool that helps developers to debug their code. The Apache open integrates easily with many services such as Kafka which is a message queuing system, Agile tools such as GitHub, JIRA, and Alarm packages which help to schedule reoccurring intervals. It also integrates with the databases such as Cloudant, push notifications. Slack messages can also be utilized in this serverless

technology provided by Apache open whisk. The official website can be found here (*Apache open whisk is a serverless, open source cloud platform*, 2021).

Apache opens whisk code to external events which can be HTTP requests like audio, video, image, a file that gets uploaded. These can be considered as external events and when invoked, trigger a function. The function contains backend logic in Apache open whisk. The deployment is containerized which makes it use of most of the frameworks such as OpenShift, Kubernetes, etc.

There are few pros of using Apache open whisk than other popular services such as Google Cloud, AWS lambda. The first and foremost is that this is open source where other services are commercial. Apache open whisk provides high security as the deployments are hosted on-prem, which makes the developers run their code in the Apache private network rather than using third-party APIs. There is no vendor lock-in, whereas other famous services suffer from this. The Apache open whisk uses various technologies in its architecture such as docker, Kafka, and Cloud DB. The Apache open whisk is most widely used for mobile applications, web applications, event-based processing IoT devices. The code that is developed by the designers can be debugged in real-time with diverse tools supplied by Apache (Shivang, 2019).

Here the functions are stateless which are later triggered by events. The application is an open whisk is a collection of actions. Action can be grouped in packages or sequences. It contains a controller that manages entities, an

invoker that launches all the actions, and the action container that executes the actions. All the actions are functional which can be invoked with input, event-driven where environments are being activated by the events, and are time-bound which means it is implemented in the quickest time as possible (Sciabarra, 2021)

2. Kubeless:

Ever since the launch of AWS Lamba by Amazon, all the significant cloud providers construct serverless technologies. Kubeless is one serverless technology developed by a Bitnami project. It is developed based upon the Kubernetes cluster by taking privileges of Kubernetes. The advantages of this are being open source, supporting multiple languages such as Python, Ruby, PHP, .net, and providing custom runtimes. It triggers events using Kafka messaging system and HTTP events. This launches run-time based upon the demand for each custom resource. It helps developers deploy code using their infrastructure and provides auto-scaling (*Kubeless*, 2021). The tools can be run locally or in the cluster. The official website to use Kubeless can be found here (*Kubeless*, 2021).

Kubeless provides write code once and runs anywhere and anytime service. Different development tools can be used with Kubeless. Kind is one such tool that helps to generate the Kubernetes cluster inside the docker container. Once the function is created, it can be deployed in a local cluster

and the function is called. Triggers can be configured using the Kubeless through the HTTP endpoints (Rojas, 2020)

The functions deployments are easy in Kubeless compared to other serverless services. The function can be involved by running one command. The script dynamically takes the contents of the file located in the /Kubeless folder and executes when a request is being invoked or sent. Changes can be quickly iterated (Okteto Team, 2020) . Kubeless provides easy operational management because the services and the applications are run on a separate infrastructure. The cost is less while using this service which uses pay-as-you-go architecture. applications run smoothly and take less operational time because of the serverless fast invocation feature. The Kubeless is used as described. A namespace is created, code is deployed in a Kubeless cluster, the use case is simulated using the CLI command, service is being accessed. The services listen to the messages and trigger the work. There are three trigger types used in Kubeless such as HTTP triggers, scheduled triggers, and pub-sub triggers(the triggers that are managed by Kafka clusters) (Singh, 2012)

3. OpenFaas:

OpenFaas is an open source serverless platform that helps to deploy functions and the code to Kubernetes. This allows the users to limit the idle time, connect with other possible resources, process data quickly and effectively. It is also capable to handle various requests that have intense processing (Ribenzaft, 2020). To write code and access applications, the users

must divide the logic into individual tasks before implementation. Complex applications can be run by simplifying serverless functions in the docker containers. The OpenFaas architecture contains an API gateway that routes all the functions, a Function watchdog that acts as an interface between the user and serverless, a Docker Swarm, and Kubernetes which are the engines to create products and local functions.

The official website to use the OpenFaas can be found here (Ltd, 2021). According to the official website, it states that the user can write, deploy their first python function in just 10 to 15 minutes. The OpenFass supplies the Templating System that lessens and communicates codes in the equipped template store. OpenFaas enterprise is available for business and provides extra functionality like the scale to zero, single-sign-on, and Kafka integrations.

As this is publicly available, they are a few pros and cons of using OpenFaas. The errors can be corrected easily while adding new functionality can be quicker and easier. It supports multiple programming languages like Java, python, c#, Go, Ruby, ASP.net, Bash, and binaries like ImageMagick, FFmpeg, etc., and users can use any based upon their convenience. The drawbacks include cold-start time for the functions to be operational in few supported programming languages. The function's life span is limited as the platform will be created and destroyed automatically. This means that stateless is not achieved (Ribenzaft, 2020). The advantage of open-source over

to the public providers is that they have vendor-lockin which will have restrictions to the function. The open source eliminates this limitation and provides on-premises deployments such as OpenFaas. This uses the HTTP and event function triggers. The developers provide functions, and the command-line interface (CLI) manages all the packages into the Docker container. OpenFaas has a flexible architecture compared to other open source serverless frameworks (Mohanty et al., 2018). OpenFaas auto scales as it uses the metrics of Prometheus which is an integral part of its architecture. This is easy to use and can be installed with one click. the functions are written in the language selected by the user and the packages are written in OCI/Docker image formats. The cold starts don't apply if a function is not scaled to zero (*Openfaas 8.0.4 · helm/openfaas*, 2021).

4. Fission:

Fission is also an open-source platform that runs based upon the Kubernetes-native serverless framework. This is developed by Platform9 private cloud provider. This allows developers to write the functions in the desired language that are supported by Fission and then map to HTTP requests. These functions can be deployed using one command. The fission automates all the configurations that help developers to focus only on coding. It uses an Apache license and works on the Kubernetes cluster. It has the flexibility of deploying the services anywhere. The Fission supports various programming languages such as Java, Python, PHP, NodeJS, GO, etc. The

official website for the Fission can be found here (*Fission*, 2020). The main edge of Fission is the low cold start time which is commonly 100msec. This is achieved by using the pre-warmed containers that run on the cluster. The developers can build functions once and can deploy anywhere using the Declarative specification feature of Fission. By integrating with Fluentd, logs will be incorporated directly into Command Line Interface. Tracking of metrics and dashboards is achieved through the integrations with Prometheus. Managing the micro-services is achieved by integrating with Istio open-source platform. The functions scale automatically bases on CPU usage.

Complex apps can be created in a simpler way and accelerated deployments which makes the applications available in just one hour using the workflows. Testing is a lot simpler as the Record and replay functions examine the function's routines. Autoscaling help achieve cost optimization. There are several use cases such as API backend that helps for web or mobile applications to write functions without worrying about servers. The Kubernetes services integrate with other services such as Redis or Postgres or Etcd. Event-driven systems are incorporated in which if any event occurs, the Fission executed the activities. This expands web applications without affecting the actual application (*serverless for Kubernetes with fission functions as a service*, 2021)

The Fission uses three main concepts functions, triggers, and environment. The function is where the code is written. The environment is the language

used and the triggers are used to execute the functions. It uses three elements that are Executer, Router, and controller. The controller contains all the three concept functions, triggers, and environments and the Kubernetes event watches. The router implements the HTTP triggers by forwarding the request and generating responses to the target functions. The executor Pool Manager and New Deploy control the functions lifecycles. The autoscaling is achieved using Horizontal Pod Autoscaler (HPA). The fission has centralized log storage. The functions run in containers; hence the logs are processed as the container logs. The main advantage of Fission over Kubeless is the low cold start time, and some advanced features such as custom workflow and phased releases (*Fission*, 2019)

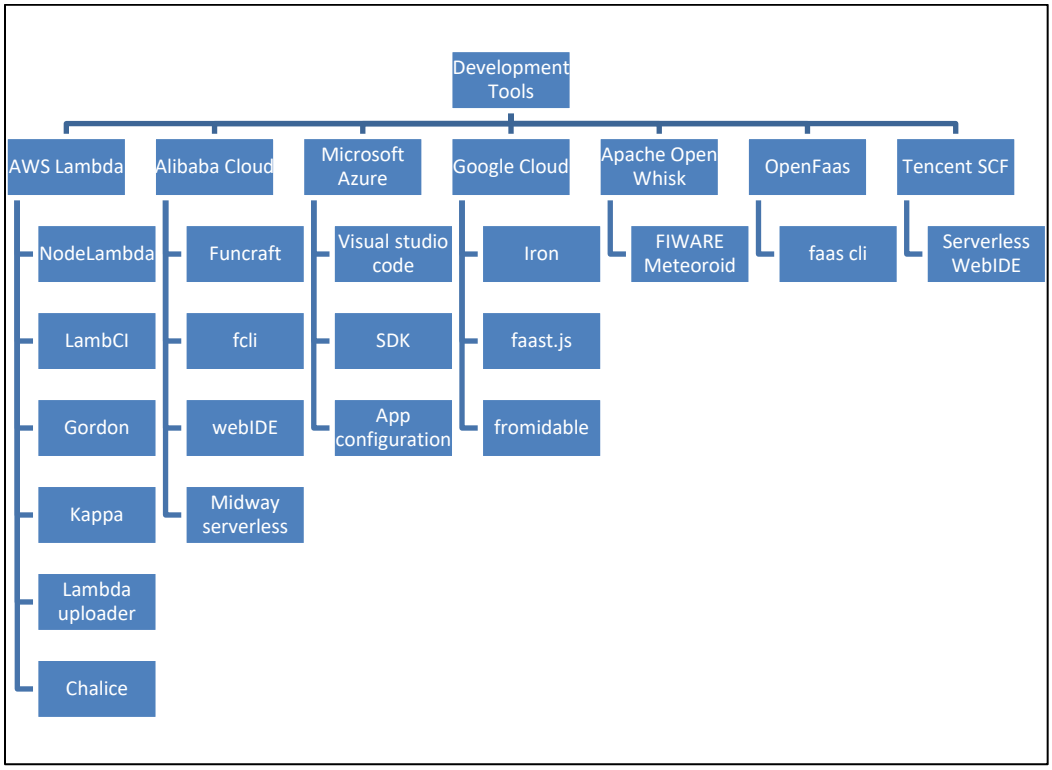
D. Features

i) Development Tools:

Serverless computing helps developers build functions without worrying about the servers. The Development tools make the FAAS services much easier. These tools reduce the efforts of configurations. The below picture depicts the tools that integrate with the various famous serverless providers.

Figure 6

Taxonomy of Development Tools



1. AWS lambda: There are many integration tools that are supported by the AWS lambda.

    a) Node lambda is an open-source command-line tool. This tool helps the JavaScript developers to run the JavaScript code locally on their machines and deploy the Node.js application into the AWS lambda. It can manage thousands of requests at the same time. Most of the developers who code using the Node.js language use Node lambda as it is very lightweight and handles all the requests instantly. The node lambda installation kit can be found her (*Node-lambda*, 2021).

b) LambCI: LambCI is a Continuous Integration (CI) tool that is maintained by AWS. This is easy to set up and quite cheaper than the other SaaS tools such as Travis or CircleCI. The code deploys easily while costing nothing. The LambCI responds to slack, GitHub, and can run using various programming languages such as Node.js, Java, Go, Ruby, etc. (Hart, 2016). The CI can be achieved through this by uploading the LambCI package to AWS. Developers can launch things and make sure that they are up to date. Code is pushed into GitHub, and it triggers the event. The LambCI tool eliminates the limitations of traditional systems. It runs around 1000 concurrent builds which in turn increases the efficiency. The main disadvantage of using this tool is that there is no root access, and it can only be supported in Linux. Memory is also limited to 1.5 GB (Taylor, 2018a).

c) Gordon: The Gordon is an open-source tool that helps to create or write or deploy AWS lambda using the cloud formation. The files can be found on this GitHub page (Bastida, 2015/2021). It allows various programming languages such as Java, JavaScript, Go, python and Scala. Gordon manages Lambda and connects the lambda to amazon s3, dynamo, and various amazon services. Gordon takes care of complex integration and makes the process smooth for the Lambda developers (*Lambdas—Gordon 0.7.0 documentation*, 2015). The process is as follows. It first downloads the requirements from the Lambda functions, zips the file, and uploads it

into the s3. Now new version can be created with the code and publish the version. IAM role is created and attached to the lambda function (Taylor, 2018b).

d) Kappa: Kappa is also an open-source command-line tool that is used for developing lambda functions. Kappa automates the process of deployment, updating, testing, configuration, adding event sources, and uploading of functions (Yegulalp, 2019). Kappa creates IAM policies, roles and helps to associate those policies with them. All the functions are uploaded in zip format into was lambda. Logs can be displayed using the cloud watch log stream. Kappa can be installed using pip and all the steps can be found in GitHub (Garnaat, 2014/2021). It uses IAM policies for access, zip functions, sends the test data, adds events to trigger the function. Changes can be made easily to lambda functions using Kappa. This framework can be used for IoT and cloud, create microservices, and dynamically wire IoT devices (Persson & Angelsmark, 2017).

e) Lambda Uploader: Lambda uploader is a command line utility that is used for python AWS lambda functions. It supports work in progress and pull requests. The lambda uploader will automatically check for requirements in configuration file with the help of requiremenst.txt file. It controls creation of virtual environment and install their dependencies. This Zip those dependencies and interface the package that in turn reduce the tasks

that are more time consuming and makes tasks easy for the developers (Rackerlabs, 2020).

f) Chalice: Chalice is an AWS serverless framework that helps to build python-based applications. Applications can be quickly created and deployed on AWS Lambda. The commands to implement the Chalice can be found in GitHub (*AWS Chalice*, 2016/2021). It acts as a command line to create and deploy applications, acts as an integration with the AWS S3, AWS gateway or any other services. This also automates the IAM policies. Not just creating, deploying but the applications can also be deleted using chalice delete command (Nayak, 2020).

2. Alibaba Cloud:

In addition to Alibaba cloud serverless, it provides few tools that help developers to run their programs smoothly and deploy their applications in local environments. All the tools can be found here (*Function compute– Alibaba Cloud*, 2021a, b).

a) Funcraft: Third-party dependencies can be installed using the fun craft. It helps to manage function compute, log services, and API gateways. This is a command-line tool and can be installed in either windows, macOS, or Linux. All the resources that help to deploy, build, or run applications can be defined in template.yml. In addition to these, code packages can be built, NAS (storage that provides access and capacity)

files can be managed and uploaded on-premises (*Features–Legacy*, 2021).

b) fcli: fcli is a command-line tool that helps to manage function compute resources. The fcli can be downloaded from GitHub (*Releases · aliyun/fcli*, 2020).There are various commands that perform various activities such as sandbox where the third party libraries can be installed to perform various operations like debugging in the local environment. service-related commands to create or update services, function-related commands to create, execute or update a function, trigger related commands to fire a trigger or update a trigger and log related commands to create Log store and run the logs (*Use fcli for the first time–Legacy. Alibaba Cloud Documentation Center*, 2020).

c) webIDE: webIDE tool that is provided by Alibaba cloud can be used to manage function compute resources.

d) Midway Serverless: This is a serverless framework that is mainly used for Node. js-based cloud functions. This reduces the maintenance price thereby focusing more on the development. Its integrated development solutions make the applications easy to deliver and maintain. It is lightweight and publicly available. This has entry and exit parameters for the function platforms. It can be migrated easily between the platforms by expanding runtime API's and unified configurations.

Integrating from traditional web to serverless can be faster using this Midway serverless framework (Midwayjs, 2018).

3. Microsoft Azure:

Microsoft offers various SDK tools that help developers build applications using various features of fully integrated development environments (IDE). It provides default azure support and advance debugging abilities. All the tools can be found in the official website of Microsoft listed here (*Developer Tools*, 2021).

a) Visual studio code: This makes developers easily switch between tools. It is used for the development, debugging, and deploying of code. All the cloud projects can be deployed locally using this tool. It also makes developers set the deployments automatically to the cloud. With the provided MongoDB support, the application data can be easily managed or hosted in the cloud using the azure Cosmo DB free tier. Easy collaboration can be achieved using GitHub which enables pair programming. Docker extensions enable containerized applications, Kubernetes extension to deploy cloud-hosted Kubernetes in visual studio code. Both the front-end and back-end applications can be debugged simultaneously

b) SDK: The SDK makes users install language that is specific to their application. They are the collection of libraries. These libraries are approachable, consistent, dependable, idiomatic, and diagnosable. This

supports various programming languages such as Java, .net, C, C++, Python, android, Ruby, IOS, PHP etc. and all the documents related to these languages can be found here (*Download Azure SDKs and tools,* 2021).

c) App configuration: All the azure app configurations can be stored here which provides a universal hosted location. It eliminates time-consuming deployments by managing the configurations effectively. This is specifically designed for speed, security, and scalability. The encryption is done for the data that is in rest or the in-transit data. It can also be integrated easily with other popular frameworks. Users will have more control and reduce costly deployments. The universal configurations make it easy to trouble shoot while eradicating the errors. The code separated from the date makes it to be a secured tool for the developers

4. Google Cloud:

a) Iron: An ironworker is a platform that is flexible to not only google but also various famous serverless services such as AWS and Microsoft azure. The main advantage of the iron platform is that it eliminates vendor locking. This helps developers to run code in various environments whether it be in the cloud or on-premises. It supports various programming languages such as Java, python, ruby,

JavaScript, Go, Node.js and .net. This was said to have the best customer support (Nick, 2021).

b) faast.js: faast.js is a framework that is multifunctional and callable as the Google cloud functions or AWS serverless functions. It ensures in uploading the code or creating the cloud infrastructures or cleaning up the code. It can scale the functions in just seconds. The fasst.js library has no operational overhead time, no service dependencies, or any complexities. This also sets up IAM roles, if the user has a google cloud provider account. serverless functions are scaled to batch jobs. this is very cost-effective and there are no clusters to manage. faast.js creates and cleans up the infrastructure when completed. This also works in local processing mode (*github–faastjs/faast.js at thechiefio*, 2021).

c) Formidable: This is a serverless module written for Node.js or JavaScript languages. In addition to google cloud, this also supports AWS and Microsoft azure. This module parses from data. This is great at error handling and provides high test coverage. It occupies low memory and allows plugin APIs in which users custom plugins and parses. This is sad to be fast and automatically write the file uploads to disk. Users have the option to use Koa package to use formidable manually (*The 50 most preferred open-source serverless tools*, 2021).

5. Apache Open whisk:
    a) FIWARE Meteoroid: A meteoroid is an open-source software that integrates firmware and Open whisk to execute the functions. users can manage the Meteoroid from the command-line interface. It is also easy to process the data. meteoroids are made of containers, which makes it instantly build the FaaS environment that can perform and execute applications. The meteoroid can be implemented either with fiware-faas-integrator or without it. This enables the developers to focus on coding while eliminating the cumbersome infrastructure preparation (*Welcome to meteoroid documentation*, 2021).
6. OpenFaas:
    a) faas cli: This is a command-line interface for the OpenFaas serverless technology that is built on docker and Kubernetes framework. Functions can be built and deployed using the faas-cli into OpenFaas. Once the function is written, the CLI does the docker image processing. It helps to build or push docker mages, deploys, removes, or invokes the functions, and manages the secrets to the functions. A single file can have multiple configuration operations using environmental variable templates. This also allows third part templates to be added to the local machines (Elli, 2017).

7. Tencent scf:
    a) web IDE serverless: The tensent scf launched the serverless web IDE for the user to have an integrated development environment for browsers. It provides an on-cloud development experience. Development, deployment, and testing of functions can be done using IDE. Code autocompletes and smart prompt options are available. It contains all the configurations that are supported by SCF such as programming languages, pip, npm, etc. Functions can be deployed and triggered either manually or automatically. This also provides an option to view the logs that include response data, output logs, and the execution summary. By default, it provides 5GB of storage space. There might be a risk of data leakages, hence they recommend installing phpMyAdmin components (*Serverless Web IDE. Tencent cloud*, 2021).

II. **Domains:**

Serverless technology is most widely used in all major domains such as IoT, Web, and mobile application domains, big data, machine learning, mathematical computations, etc. Web services is a dominating domain in serverless technology. Next comes scientific programming and IoT. Popular services such as Slack, Netflix, Coco-cola use serverless technology. Netflix uses AWS lambda to mage the infrastructure using event-based triggers. Shamrock's trading company services are fully (100%) serverless. They shifted from docker to serverless which made them save a lot of money (David, 2021). Web applications can be built

easily using serverless. it ensures scalability, cost-effectiveness, and quick building of applications. According to one survey conducted by O'Reilly, about 40% of the respondents have adopted the serverless architecture in their fields in various industries such as software ranking 1st and follows with finance, consulting, telecommunications, health care, government, and manufacturing (Guzikowski & Chris, 2019). This survey includes that AWS lambda was the top service to be used by followed Microsoft, Google, IBM, and Oracle. The tools that were most frequently used were Custom tooling, Cloudflare, lambda uploader, node-lambda, LambdaCI, Cloud Zero, and Kappa. Although there are few drawbacks considering security, vendor lock-in in which few companies were not very enthusiastic to adopt the serverless technology, most of the companies did.

Big data contains a large volume of data that is unstructured. The data increase daily and maintaining od such large data and performing analytics on such data can be quite complex and high in cost. Using serverless architecture in big data analytics can make the tasks easier. Implementation, maintenance, and governance of these applications in the serverless can be effective (Rahman & Hasibul, 2019).

Machine learning systems typically use the serverless architecture as the data increased in volume and is difficult to handle. Stateless functions are being executed in the cloud without worrying about the infrastructure and their maintenance. This dynamically controls the memory of the stateless functions

(Wang et al., 2019).The serverless architecture has components in which all te common functionality is located. For example, plug-and-play is a component that is used to structure and sort the data, and testing can be done easily on this. It also reduces the development resources barriers which in turn reduce the infrastructure costs. The serverless automates many of the big data or machine learning challenges. vendor-native comments is just what needed to switch to serverless architecture and is compatible with any of the major serverless providers (Adam, 2019).

Internet of things is rapidly adopted by many companies and provided various technological innovations. Implementing those IoT devices using the serverless technologies lowers the cost of infrastructure while focusing on the output or the application. For example, if an IoT developer tries to utilize the was services, they use AWS IoT of integration, Dynamo DB for storing the data, and AWS lambda for processing of the data. Once the function triggers, the results are achieved by calling the data that is stored in the tables. Hence, this reduces operational, developments and deployment costs. It is fault tolerant and is quite scalable by default (DataArt, 2017). AWS lambda and other public provider renders huge support for the services that helps to build secure, reliable IoT platforms. Developers can create the IoT backend to collect the data, visualize and analyze the data.

Serverless architecture has the capacity to keep complex configurations uncomplicated. Serverless can be used to access numerous CPUs at low costs.

Mathematical computations such as linear algebra can become complex if it is being one machine or of larger calculations. High-liner algebra algorithms can be implemented using serverless architecture. According to the (Shankar et al., 2018), LAmbdaPACK package can be used to achieve this. it is 240% better because of its elasticity and can perform matrix and other complex mathematical operations. Functions as a single core in serverless systems which makes them execute on any machine. This shows that the algebra algorithms can be executed with stateless functions and with seamless fault tolerance. the elasticity in the serverless technology enables the system to dynamically adjust to the natural linear algebra algorithms.

III. **Service Integrations:**

While using services such as AWS Lambda or Google cloud, the developers have to take care of the errors that they encounter. hence, they try to modify the business logic in their code. By integrating with the API gateways, they can improve the reliability of the application by reducing the number of lines in the code. At first rest API's were being used for integrations. Now HTTP APIs were introduced with fewer configurations than the rest API and have automatic deployments. The HTTP API directly integrates with the services offered by Amazon. without any templates. The direct HTTP API integrations are AWS system manager, step functions, event bridge, data streams, and simple queue services.

The System manager provides the interface in which users can track the operational issues and resolve them. Tasks can also be automized. It helps to monitor,

implement and troubleshoot the application issues. This integration is built to simplify the application management and lesser the time to resolve the operational problems. This is very helpful to automate workflows, runbooks, and rollout in case of errors. It helps to achieve greater control over the applications. The agile issue or requirements tracking tools such as ITSM or service now can be integrated easily with this (*AWS systems manager – Gain operational insights and take action*, 2021).

The AWS step functions help to visualize the workflows, automate the processes and build the applications using serverless technologies. It helps developers to focus on codes while this service integration takes care of failures, integrations, parallelization, etc. This automates the manual jobs of Extract, transform and load (ETL) processes. It also makes the data easily processes for machine learning. It combines several functions into microservices and serverless applications. There are various industries that use the step function services (*AWS step functions. Serverless microservice orchestration*, 2021).

The Amazon event bridge helps to build event-driven applications at scale. It takes the data from the Zendesk or shopify. Rules can be arranged to verify if the data reacts with the real time data sources. It is beneficial to connect the SaaS applications without writing code. It has fault-tolerance and built-in distribution availability. This connects applications using custom events. Monitoring and auditing of the application is possible in the real-time to prevent vulnerabilities (*Amazon Event Bridge*, 2021).

Amazon Kinesis Data Streams (KDS) is used for data streaming in the real-time. It seizes data from multiple sources such as social media feed, logs, financial

transactions, location tracking etc. This data is useful for real-time data analytics and enables dynamic pricing, anomalies detection. It is secured as data is accessed through virtual private amazon cloud and easy to use as data processing is quick, i.e., within 70 milliseconds form the time data is collected. It is durable and ensures that there is no data loss and provides multiple layers of protection. It is high in elasticity as it dynamically scales up the applications and is very low in cost. This is used in gaming, mobile applications, event data collection and in analytics (*amazon kinesis data streams–Data streaming service*, 2021).

Amazon simple queue services (SQC) enables send, receive, delete, or prioritizing messages using HTTP API direct integration. This integration assures that the data is secure through the encryption of logs. It scales automatically and other is no limit for the messages in the queue. It provides reliable message delivery and eliminates infrastructure overhead. This service is used in IoT, healthcare, and various major industries (*Amazon SQS. message queuing service.* 2018).

Hence these are some of the important service integrations that t are provided by the major cloud provider amazon that helps to easily integrate with the serverless technologies to make the work of developers easy and convenient.

***IV*. Programming Languages:**

Serverless supports many programming languages for their developers. They can choose a language based upon their choice to build any applications in the serverless platform. For example, AWS supports C#, Java, Python, Node. js, Ruby, PowerShell and Go. They also enable runtime API which makes them use additional

programming language. The cloud run supports more languages and allows it to run on containers. Python and Node.js are the most popular languages in serverless computing. Node uses express pattern, Python used flask, Go used Http packages, Java uses Maven and it also supports other Java Virtual Machine (JVM) languages such as Kotlin, Scala, and groovy. The .net is built upon ASP.NET core, Ruby on the standard ruby interface, and PHP using PHP server (Timmerman, 2020).

V. **Scalability:**

Another factor that makes developers prefer serverless computing in their applications is its scalable ability. Applications that are built with a serverless infrastructure tend to scale automatically as the user base grows which intend means usage increases as well. Here when functions need to run, they do so in multiple instances, and all this is done by the provider. They start to run and end as at when needed with the use of containers. Using containers means the functions start up much quicker especially when they have been run not too long ago. This means that it is highly impossible for a serverless application to be overwhelmed in case of an increase in usage as they can handle a huge number of requests and also process a single request as compared to a traditional method which has a fixed amount of server space.

VI. **Reduced Cost:**

With serverless computing, developers are only charged for what they use this is so because code only runs when backend functions are needed by the serverless application, and because of this, the code automatically scales up as looked-for.

Providing users with access to applications is dynamic, accurate, and real-time. In comparison to traditional methods, a developer had to project how much server size they will need and then acquire that capacity in advance, whether they end up using it or not. This is a very huge factor that has led to many developers switching to serverless as it resembles pay as you go in the phone industry.

VII. **Decreased Latency:**

Latency means delay. Codes can be run from anywhere since applications are not hosted on the origin server in serverless. This makes it possible for functions of applications to run close to the user reducing latency because requests from the users don't have to travel all the way to the origin server. This is another factor that makes developers choose serverless computing over traditional methods.

VIII. **Quick deployments and updates:**

Developers prefer using serverless for their applications because quick deployments and updates are possible. Serverless infrastructure makes it possible to release a working version of an application without uploading code to servers. This makes it possible for developers to release their new products, release code in bits, upload code a function at a time or all at once. Developers can update, fix and add new features to their applications a function at a time rather than make changes to the whole application.

E. **Vulnerabilities**

I. Vendor-lock in:

Vendor-lock in is a situation in using the same vendor products. If a user wants to shift to a different vendor, it costs very high. Hence it is said that the customer is locked into the service provider. Serverless has the vendor-lock in issue as it makes users lock to its vendor. It has potential risks, for example, the vendor controls the host, resources such as hardware and operational systems. The API acts as an interface between the user and the provider. The API can become vulnerable, prone to attacks as it is maintained by a third party. This in turn affects the users that use serverless technologies.

Tight coupling in the code will make it difficult for the user to shift to different vendors in the future (Vijayan, 2021). Sometimes vendors may change the product that might not support the business need of a user. As the clients are already locked in there are chances of increasing the price of their service. It also increases the complexity of the maintenance.

Solutions to avoid vendor locking will be the cross-provider deployments, which help to deploy the same code on different serverless platforms (Taibi et al., 2021). Some open-source FaaS frameworks are suggested to overcome this. Multi cloud approach can be followed to leverage this problem, ensure probability and utilize serverless computing effectively (Sampe et al., 2021). Using the programming language that is supported by any of the providers makes it easy for the user to switch from one vendor to another vendor. Migrating to the vendor that supports the existing programming language would reduce the migration costs. Users should consider the architecture pattern. A good pattern makes the

migration easier for the clients. Abstraction plays a role like Operating systems abstract from hardware, virtualization abstracts from Operating systems, similarly architecture pattern in serverless (Tanasa, 2019). Lastly using the technologies that are standardized makes the migratory easier which reduces the vendor-lock in problem, for example, HTTP webs server is supported by almost all the serverless vendors. Hence, migration will be quite easier in such cases. SQL database is one standardized technology that can be integrated with any of the service providers. Hence, keeping in mind with these few techniques reduces the vendor-lock in problem in serverless technology.

II. ***Cold Starts:*** Cold start is expressed as the response period it takes when the instance starts a new request for the first time after deployments. Cold start is one most common problem in the serverless world. The reason for the request to process longer is that it needs to initialize the worker, function module, and then allocate the worker(Cui, 2020). Hence, this in turn results in more time for the containers to warm up and start the services. It might take some duration for the code to execute. The cold start can occur depending upon the language we use, the dependencies, etc. The scripting languages such as JavaScript, Python, Golang, and Node.js have better performance than Java, C# and .net.

According to the *Comparison of Cold Starts in Serverless Functions across AWS, Azure, and GCP* (Shilkov, 2021), the azure functions take 20 to 30 mins, AWS lambda consumes 5 to 7 mins and the Google cloud consumes 15 mins.

Hence AWS has fewer cold starts compared to Azure or google cloud. The packages inside the languages such as java, JavaScript can increase the cold start.

The cold starts can be mitigated by taking several measures. Sometimes allocating more memory to the functions help to startup fast. Removing unnecessary packages or shrinking the package size before deploying helps in improving startup latency. Keeping the functions warm reduces the cold start frequency. This can happen by invoking functions periodically, which reduces the warmup time. Large functions lead to more cold starts. hence, breaking down the function and having proper architecture increases the performance (Rehemägi, 2021a).

III. Less Control:

Serverless relies on third parties in several ways such as libraries, software, or API. Such dependency leads to less control of the system and more vendor control. If the vendor has any downtime or vulnerabilities or cost modifications or upgrades, it directly impacts the users that are using serverless which are dependent on those vendors. Even if the user needs to switch from one vendor to another, it would be very difficult as they need to change all the configurations that need to be adjusted as per the new vendor, modify the logic and change the designs and architecture. Hence, it causes a lot of complications to the developers. Even with all the changes, the components might get affected. Each of these vendors might have different security concerns which directly impact the serverless users (Roberts, 2018).

One main advantage of serverless is that we can customize the software, version, packages, dependencies, etc., as per the business. Hence, the software is vast and will have less control over all the vendors and it would be difficult to track all the stacks. The loss of control can be within performance or configurations or problems resolutions. Configuration-wise, the serverless allows to configure as per their want but leads to less control and exposure of data by the vendors. The vendors provide tools that are more platform-specific rather than operating-system-specific. Hence, they can be less compatible with the Operating systems (Roberts & Chapin, 2021a).

This can be reduced by using risk assessment and use what are the important features for the business. This will have more control and conduct less effect. Tools such as Dashbird helps to analyze the metrics, logs events, monitor the errors and health of the lambda functions. It also specifies memory usage and avoids cold starts. Users can know the troubleshoot the performance issues using the Dashbird Tool (Rehemägi, 2021a, b, c).

IV. Difficult to debug:

Debugging is a bit difficult in serverless as it has distributed components which make the live environment debugging difficult. It is not even cost-effective. Users will not be able to debug live as they do not have direct approach to the servers. Since debugging is live, it costs more that might directly impact the business budgets. Few of the frameworks that the serverless applications use have limitations imposed which makes debugging quite difficult. Tracing and tracking

the code is hard. Sometimes debugging in serverless computing uses entire logs that make debugging difficult.

The debugging in the serverless world is divided into two areas such as local debugging and debugging serverless in the cloud. Frameworks such as AWS Lambda use Serverless Application Model (SAM). This YAML template helps developers to define events, functions, and permissions. The command-line interface helps to invoke functions locally and debug them (*serverless debugging guide*, 2022).

Debugging in the cloud requires a few connections that help for live testing. The First would be testing APIs such as postman and curl. secondly, testing the lambda functions where test events are created. track those applications using AWS cloud watch logs using familiar tools such as Splunk or ELK. AWS X-RAY to debug and analyze the applications. that helps in debugging (Sengupta, 2021).

V. Low privacy:

Considering all the points mentioned earlier, serverless is prone to security vulnerabilities. The main reasons can be due to using multiple resources that tend to make data more visible and less secured. The attack surface can be more due to numerous providers and various services. The data in serverless is stored in stateless mode. This can cause privacy issues while transferring the data from one location to another and can lead to data leaks. Insecure settings in the deployments can lead comprise the privacy of the files.

Hence it is recommended to follow best practices while using serverless technologies or applications. Always secure the data storage, Cautious with third-party services, follow best coding practices, and only use the trusted 3rd party libraries that can help in mitigating security issues in serverless.

F. Use Cases

Serverless technology can be used in a variety of applications or systems which are discussed below. Most of the applications can be built upon the serverless with fewer limitations. It has fine grade deployments, less monitoring or maintenance, and more of focusing on development tasks. Below are the most popular use cases of serverless computing.

I. IoT:

The most popular Alexa, iRobot is serverless users. This itself shows how serverless is very prevalent among the household appliances or day-to-day things that build using Internet of Things (IoT) technology. The IoT is restructuring its models and adapting to serverless computing since it resolves many of the IoT problems such as sustainability (Yusuf, 2019).

IoT is expected to have a greater impact in economically poor societies. But it comes with more security vulnerabilities. IoT in public domains contains sensitive data and has a more prominent chance of exposure or a breach that leads to security-related issues. This is why few countries don't adopt IoT specifying the major security concerns. IoT also needs technical infrastructure such as cloud,

network, architectures to implement the solutions. All these issues pushed back IoT.

Serverless came into the picture which made IoT publishers solve most of the above-mentioned issues. AWS, Lambda solutions in IoT helps to maintain data privacy even in the public domains. The Function as a service (FaaS) and Software as a service (SaaS) solved economic issues of IoT. The serverless architecture is built in such a way that it addresses all the problems and makes IoT easier and more secure to be used. It reduces maintenance costs, hence can be utilized in economically weaker countries. Network, cloud, and data processing are made easier if IoT uses serverless technologies for their backend processing. Nevertheless, it is cost-effective which advantages IoT users and publishers. They provide quality, secured effective services and monitoring can be very easy for the IoT appliances that are built upon serverless technology.

II. Continuous Integration:

Users or developers spent most of their time on Continuous Delivery or continuous Integration known as (CI/ CD) integration in addition to developments and deployments. Suing serverless technology helps them to focus mainly on the developments while the CI/CD is taken by serverless. It is easy to set up and maintain. Once the pull request has been created, the changes will be deployed automatically, which helps the team or the developer to test, preview or view the outcome (*Serverless CI/CD: Built for serverless applications*, 2022). AWS is built

in such a way that it can easily integrate with GitHub and helps in serverless CI/CD

According to the (Mishra, 2022), there are several uses for CI/CD that use serverless or FaaS technology. Server maintenance is not bothered, and there is no cost for idle time. It can have several builds happen concurrently which can be more than 100 builds. The only drawback is that its support for plugins is limited and there is no root access. Automatic deployments and builds have been fostered. Custom pipelines can be created for highly developed deployments.

III. Data Processing:

Serverless is highly used in various types of data processing such as image, audio, video, text files, PDF processing, multimedia processing, etc. It is known best for Ad Hoc Data Processing. Frameworks that automatically deploy helps in storing data quickly in a cloud environment. There are yet a few things that affect the data processing in serverless such as the size of the files, the way files are split, etc. (Werner et al., 2020).

Amazon uses Kinesis for data processing. Firstly, the data such as financial or logs or IoT data is being sent to Amazon Kinesis. This data is stored in the form of a shard. The Fanout technique is used where the lambda functions are triggered and the date is pushed out. The data is thus stored in Dynamic DB. It uses a push and pull model, where data is pushed from sources to the lambda functions. In the pull model, the data is mapped by the function (Sitapara, 2018).

Example:(*SiteSpirit,* 2019) IBM's "Site spirits" is a company that creates responsive websites for various industries. It uses serverless Media-library-as-a-service which is also known as "Media Spirit" to create visually effective pictures for marketing. Using serverless technology, it retrieves the right picture from thousands of images easily. It is 10 times faster and 10% more cost-effective after using serverless. Initially, it was developed using an open-source database and run on virtual servers using Platform-as-a-service technology. Now they have completely moved to serverless, changes the architecture, and started using Open Whisk, IBM's serverless technology. This is now handling its high-demand requests and delivers quicker responses.

IV. Web and Mobile applications:

Serverless technology build backend APIs that services mobile applications and web apps. The Application developers tend to move to serverless as they can just focus on coding and logic, while the serverless take care of all the processing, backend data, monitoring, storing, etc. This results in a quality product, where their focus will be more on functionality that matches their business. Whenever developers or serverless users wish to perform changes to their functionality, they can edit only the function rather than changing all the backend which in turn makes easier updates. It is cost-effective as they only pay for what they need like resources that are being used, storage space, etc.

Serverless also offers predefined frameworks, which make the life of developers so easy by just editing in the given frameworks. In FaaS, the functions

run independently where the functions are divided into small functions. whenever the user triggers the event, the function runs, and the cost is calculated based upon the time the function is running. we only pay for the running time and not the idle time. Deployments are quite easier using serverless. There are many providers available, and the users have options to choose from a wide variety of cloud suppliers such as Google cloud functions, AWS, Apache, etc.

Below are a few well-known companies that use serverless technologies: (Arkhipov, 2020).

1. T-Mobile: This company use serverless to build their mobile and web applications.
2. Coca-Cola: These companies use vending machines that are implemented using serverless and gained a lot of popularity.
3. Slack: This uses serverless applications for notifications.
4. UPS: This company uses serverless technologies for their communicating bots. UPS uses Microsoft Azure serverless services and only focuses on client-side functionality while the backend is all taken care of by Microsoft serverless.

V. Micro Services:

Microservice means an application is divided into smaller parts and all these parts execute separately rather than the application being executed as one rigid. These have separate operations, data storage, and services. Microservices use serverless deployment models for their deployments as they are fast, highly

elastic, and at lower costs. Monitoring and running microservices can be taken care of by serverless providers.

Using serverless, the microservices only run when needed by applications. It can also de further divided into smaller functions depending upon the complexity (*Tozzi*, 2021). Code can be written in microservices, and they can run upon serverless. The code is executed when they receive HTTP requests and provides a response. The developers need not worry about the infrastructure as the serverless providers take care of that. This improves efficiency and low overhead time. The simplicity makes the complex applications manageable, scalable, and easy to implement.

The fewer challenges with the serverless microservices are that cold starts which reduce the function performances. As every application consists of many microservices, it will be a bit hard for monitoring all those services. It might be hard to determine the root cause for the issues as sometimes it can be due to function timeouts or can be due to the environment (Datadog, 2020)

Considering the major positives of serverless microservices, it is the best way to integrate them as they both have numerous advantages.

VI. Cloud Automation and CRON jobs:

Cronjobs are used by almost all companies to automate cleanups, email notifications, etc. Serverless technologies such as AWS Lambda can be used to perform such jobs which reduce machine management and have very simple interfaces that are used for automating tasks (McCumskey, 2019).

This is cost-effective because serverless is paid as you go. Hence when crone jobs are scheduled to run at a particular time, the serverless only costs for that, and the rest of the idle is not calculated which can be a great benefit for the companies to perform crone jobs. Additionally, it also establishes anomaly detection, monitoring, and metrics.

To have cloud automation, first, the production database needs to be connected to the function where it executes all the tables and data. Next, all the dependencies and the serverless framework will be installed where the cronjobs are scheduled. AWS also provides secret management and a various number of alerts that are built in previously. Periodically backing up data can be easy using serverless as it doesn't require the server to run continuously (*Serverless advantages and use cases*, 2022).

There are very few challenges in executing cronjobs such as using different hardware such as GPU if the cronjobs execute more than 15 minutes long.

VII. API backends:

Serverless can work on Rest API backends. The web action which is enabled from the web help to assemble the Application Program Interface (API) with the API gateway. The actions which are also called functions are converted to HTTP endpoints. These endpoints are readily consumed by the clients (IBM cloud education, 2021b).

The IBM cloud function...demonstrated how this can be achieved using Open Whisk. First, the database is created such as Mongo DB, then the serverless

actions are registered which run using Function as a service (FaaS). All the entries from the database are retrieved using a sequence of actions and the database connections are being established. By the end of this, the functions will have endpoints. Now API is created, and web application is deployed (*Mittal*, 2019).

Data Analysis

The data is analyzed based upon the multiple factors that affect serverless by defining them as attributes in taxonomy. Each attribute helps readers understand the evolving technology, the precautions that need to be taken before or while using Serverless computing. Each attribute is then subdivided that explains the developer tools, which in turn helps readers understand the various options that are available and the providers that issue them. Taking all the security attributes into consideration, the developers can build their seamless applications in a secured and cost-effective manner.

Summary

This chapter educates users by understanding the features of serverless technology, attributes that need to be considered while working with serverless computing. Risks and precautions are advised that helps readers use the technology in an effective and secured manner. The data collected from several research papers, journals, and real-time incidents from newspapers helps to educate the readers and is a single go-to place to refer to all the attributes of serverless computing. The next chapter describes the results of the study, conclusion, and future work.

Chapter V: Results, Conclusion, and Recommendations

Introduction

In this chapter, I am going to design a taxonomy based upon the multiple factors that cause security issues in serverless computing. Based on the design, I will explain each branch of attributes, define the attribute, and explain why it is important.

Results

Based on the research conducted throughout the study, all the questions that were identified during the methodology has been found and addressed below.

1. How to detect and identify vulnerabilities in serverless computing?

   This research question is answered in the first attribute of the taxonomy, i.e., "Attack". It describes the Types of Attacks and the Root causes. This helps readers to detect and identify vulnerabilities.

2. Different problems are mentioned in the previous chapters. How can we drive solutions to the mentioned problems?

   This research question is answered in two branches of the taxonomy. One "Vulnerabilities and Mitigations" explains the problems and solutions to the emerging problems. Another branch "Protection against attacks" describes how to protect the serverless systems, when the problems are being identified.

3. There are different tools available such as Open-source tools like SecLambda. What are the ways to protect those tools?

   This research question is answered in the branch "Features". This explains all the Developer tools that are available which are provided by the serverless providers that

were defined under "Platforms". It also talks about the ways to protect the tools and use them in a secure way.

4. How to protect the systems that use serverless technologies?

   This research question is answered in both "Attacks" protection against the attacks and in the "Features" development tools. Also, the overall paper gives an overview for the readers to help them know how they can protect the serverless technology-based systems.

5. How should users or developers take care while using the serverless technology?

   The "C.I. A Triad" (Confidentiality, Integrity, and Availability) is one of the taxonomy branches that describes the best practices that developers should follow which in turn help them build secured applications on the serverless. "Use cases" which explains several properties of serverless computing, also describes how developers should use this technology and what precautions need to be taken care of.

6. What are the Security and privacy issues concerning serverless technology?

   This research question is answered in all the chapters of the paper. This paper primarily focused on security and privacy issues that concern serverless technology. By end of the paper, users will get a good knowledge of serverless technology, security issues in the technology, and how to overcome those issues to build secured applications. In Chapter 4, Taxonomy the "Attack–Type" defines what are all the attacks in the serverless world, the root causes of those attacks. "Vulnerabilities" in vulnerabilities and mitigations also define the security concerns in serverless technology.

Conclusion

This paper discusses the emerging technology of serverless computing and its characteristics. As there are many benefits with serverless, there are some catches that are clearly described in this paper. Research has been conducted based upon several articles, recent blogs, google scholars. The data is analyzed, and the taxonomy has been classified. The taxonomy evidently shows the features of serverless computing, the platforms that it is built upon such as AWS or Google, various attach types, and the root cause for those attacks. lastly, several use cases have been depicted such as serverless being used in IoT, data processing, microservices, etc.

In this paper, the merits of serverless computing alongside the pitfalls have been shown. A great deal of responsibility lies on the shoulders of the users/developers using serverless technologies, especially when it comes to securing their work. Users/developer must carefully protect their environment, especially from input-oriented attacks but also lapses in maintenance that can open gates for malicious elements.

Future Work

In the paper, we have shed detailed light on Serverless computing as it relates to the security of applications and the application development process, highlighted the security attributes to keep an eye on, and recommended techniques to help mitigate breaches that might be caused by overlooking these attributes.

Further research can go in the direction of making an automated tool based on the attributes we have highlighted. A tool that can scan a serverless environment and determine how much risk is presently based on the attributes we have lined up. The field of serverless

computing is relatively new, which means it is going to keep expanding. Researchers need to keep an eye out for other indicators of compromise as the serverless train chugs ahead.

References

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

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

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

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]. http://arxiv.org/abs/1802.08984

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

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

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

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

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

Apache OpenWhisk is a serverless, open-source cloud platform. (2021). Retrieved October 3, 2021, from https://openwhisk.apache.org/

Arkhipov, A. (2020, July 14). *How to build apps with serverless architecture.* TechMagic. https://www.techmagic.co/blog/how-to-build-apps-with-serverless-architecture/

Aruljothi, V. (2021). *Implementing serverless node.js functions using Google Cloud*. Toptal Engineering Blog. Retrieved October 1, 2021, from https://www.toptal.com/nodejs/serverless-nodejs-using-google-cloud

AWS Chalice. [Python]. (2021). Amazon Web Services. Retrieved September 19, 2021, from https://github.com/aws/chalice (Original work published 2016)

AWS Lambda–Serverless compute—Amazon web services. (2021). Amazon Web Services, Inc. Retrieved September 19, 2021, from https://aws.amazon.com/lambda/

AWS step functions. Serverless microservice orchestration. (2021). Amazon Web Services, Inc. Retrieved October 17, 2021, from https://aws.amazon. com/step-functions/

AWS systems manager–Gain operational insights and take action. (2021). Amazon Web Services, Inc. Retrieved October 17, 2021, from https://aws.amazon.com/systems-manager/

Azure serverless. Microsoft azure. (2021). Retrieved September 19, 2021, from https://azure.microsoft.com/en-us/solutions/serverless/

Baisakhiya, N. (2017). *Cloud bleeding-typos leaking your information,* p. 1.

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

Bastida, J. (2021). *Jorgebastida/gordon* [Python]. https://github.com/jorgebastida/gordon (Original work published 2015).

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

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-serverless-tokenization-solution-to-mask-sensitive-data/

Budzoń, P. (2017, April 5). *Intrusion detection and prevention with AWS Lambda and DynamoDB streams*. Retrieved January 17, 2022, from https://mysteriouscode.com/blog/intrusion-detection-and-prevention-with-aws-lambda-and-dynamodb-streams/

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/

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-repos-have-leaked-api-or-cryptographic-keys/

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

Cimpanu, C. (2021). *A crypto-mining botnet is now stealing Docker and AWS credentials*. ZDNet. Retrieved February 24, 2021, from https://www.zdnet.com/article/a-crypto-mining-botnet-is-now-stealing-docker-and-aws-credentials/

Cloud, P. A. (1998). Product introduction. In *Engineering Procedures Handbook* (pp. 10–18). Elsevier. https://doi.org/10.1016/B978-081551410-7.50006-4

Cloud workload protection platform. Prisma. (2020). Palo Alto Networks. Retrieved March 16, 2021, from https://www.paloaltonetworks.com/prisma/cloud/cloud-workload-protection-platform

Conger, K. (2017, February 24). *Major cloudflare bug leaked sensitive data from customers' websites.* TechCrunch. Retrieved June 5, 2022, from https://techcrunch.com/2017/02/23/major-cloudflare-bug-leaked-sensitive-data-from-customers-websites/

Cui, Y. (2020, September 20). *AWS Lambda cold starts: Solving the problem*. Lumigo. https://lumigo.io/blog/this-is-all-you-need-to-know-about-lambda-cold-starts/

DataArt. (2017, July 28). *Should you use serverless architecture for your IOT solution?* IoT for All. Retrieved June 16, 2022, from https://www.iotforall.com/serverless-architecture-iot-solution/

Datadog. (2020). *Serverless microservices: What it is & how it works.* Datadog. https://www.datadoghq.com/knowledge-center/serverless-architecture/serverless-microservices/

David, D. E. (2021, May 14). *Unfolding serverless: What is serverless and why it matters*. Retrieved October 15, 2021, from https://www.antstack.io/blog/unfolding-serverless-what-is-serverless-and-why-it-matters/

Developer security: Develop fast. stay secure. (2022, June 8). Snyk. Retrieved June 16, 2022, from https://snyk.io/

Developer tools. (2021). Microsoft Azure. Retrieved October 14, 2021, from https://azure.microsoft.com/en-us/product-categories/developer-tools/

Disaster-recovery-introduction. (2019, September 6). IBM. Retrieved June 19, 2022, from https://www.ibm.com/cloud/learn/disaster-recovery-introduction

Download Azure SDKs and tools. (2021). Microsoft Azure. Retrieved October 14, 2021, from https://azure.microsoft.com/en-us/downloads/

Duc, H. N. (2019, August 16). Going serverless? Common serverless security issues and best practices by Aaron Chichioco. *Hakin9-IT Security Magazine*. https://hakin9.org/going-serverless-common-serverless-security-issues-and-best-practices/

Elli, A. (2017, September 5). *Morning coffee with the OpenFaaS CLI*. Alex Ellis' Blog. Retrieved June 19, 2022, from https://blog.alexellis.io/quickstart-openfaas-cli/

Features—Legacy. (2021). Alibaba Cloud Documentation Center. Retrieved October 14, 2021, from https://partners-intl.aliyun.com/help/doc-detail/140283.htm

Fission. (2019). *Fission*. Retrieved October 12, 2021, from https://fission.io/

Fission. (2020). *A deep dive into serverless Kubernetes frameworks (2)*. Alibaba Cloud Community. Retrieved October 12, 2021, from https://www.alibabacloud.com/blog/fission-a-deep-dive-into-serverless-kubernetes-frameworks-2_594902

Function compute—Alibaba Cloud. (2021a). Alibaba Cloud. Retrieved October 3, 2021, from https://www.alibabacloud.com/product/function-compute

Function Compute—Alibaba Cloud. (2021b). Alibaba Cloud. Retrieved October 14, 2021, from https://www.alibabacloud.com/product/function-compute

Garnaat, M. (2021). *Kappa* [Python]. https://github.com/garnaat/kappa (Original work published 2014)

Gillas, G. (2019, February 28). *Injection attacks: Protecting your serverless functions*. Stackery RSS. Retrieved June 16, 2022, from https://www.stackery.io/blog/protecting-serverless-functions/

GitHub—Faastjs/faast.js at thechiefio. (2021). GitHub. Retrieved October 15, 2021, from https://github.com/faastjs/faast.js

Google Cloud. (2021). *Knative*. Retrieved September 30, 2021, from https://cloud.google.com/knative

Gursimran, S. (2020, October, 20). *Azure serverless computing—Architecture, advantages and tools*. Retrieved September 19, 2021, from https://www.xenonstack.com/blog/azure-serverless-computing/

Guzikowski, R. M., & Chris. (2019, November 12). *O'Reilly serverless survey 2019: Concerns, what works, and what to expect*. O'Reilly Media. https://www.oreilly.com/radar/oreilly-serverless-survey-2019-concerns-what-works-and-what-to-expect/

H., N. (2019, November 8). *6 key benefits of serverless architecture and technology*. Geniusee. Retrieved June 19, 2022, from https://geniusee.com/single-blog/serverless-architecture

Hart, M. (2016, August 15). Introducing LambCI—a serverless build system. *Medium*. https://hichaelmart.medium.com/lambci-4c3e29d6599b

Heath, N. (2019). *Writing serverless code: The programming languages and everything else you need to know*. ZDNet. Retrieved February 23, 2021, from https://www.zdnet.com/article/writing-serverless-code-the-programming-languages-and-what-else-you-need-to-know/

IBM Cloud. (2021). *DNA IT solutions*. IBM Private Cloud. DNA IT Solutions Cloud Experts. Retrieved October 3, 2021, from https://www.dnait.ie/cloud-services/serverless-computing-with-knative-in-the-ibm-cloud/

IBM cloud education. (2021a, October 8). *Serverless*. https://www.ibm.com/cloud/learn/serverless

IBM cloud education. (2021b, October 8). *Serverless*. https://www.ibm.com/cloud/learn/serverless

IBM Cloud Foundry—Overview. (2021, February 11). Retrieved June 5, 2022, from https://www.ibm.com/cloud/cloud-foundry

IBM Cloud Functions—Overview. (2021, July 20). https://www.ibm.com/cloud/functions

Intelligent threat detection—Amazon GuardDuty. (2022). Amazon Web Services, Inc. Retrieved January 17, 2022, from https://aws.amazon.com/guardduty/

Jegan, D. S., Wang, L., Bhagat, S., Ristenpart, T., & Swift, M. (2020). *Guarding serverless applications with SecLambda*. ArXiv:2011.05322 [Cs]. http://arxiv.org/abs/2011.05322

Jindal, A., Gerndt, M., Chadha, M., Podolskiy, V., & Chen, P. (2021). Function delivery network: Extending serverless computing for heterogeneous platforms. *Software: Practice and Experience*, spe.2966. https://doi.org/10.1002/spe.2966

Johnson, J. (2019, April 30). *Serverless scripting for URL redirection*. stackpath blog. Retrieved June 16, 2022, from https://blog.stackpath.com/serverless-redirect-script/

Khani, A. (2021, March 11). *Multi-Cloud security best practices for serverless functions*. Cloud Health by VMware. Retrieved January 30, 2022, from https://www.cloudhealthtech.com/blog/cloud-security-serverless-functions

Kovas, E. (2020, October 19). Google targeted in record-breaking 2.5 Tbps DDoS attack in 2017. *SecurityWeek.com*. Retrieved March 3, 2021, from https://www.securityweek.com/google-targeted-record-breaking-25-tbps-ddos-attack-2017

Kubeless. (2021). Retrieved October 3, 2021, from https://kubeless.io/

Lambdas—Gordon 0.7.0 documentation. (2015). Retrieved October 14, 2021, from https://gordon.readthedocs.io/en/latest/lambdas.html

Ltd, O. (2021). *Home*. OpenFaaS-Serverless Functions Made Simple. Retrieved October 7, 2021, from https://www.openfaas.com/

Ltd, P. (2018a). *PureSec reveals that 21% of open source serverless applications have critical vulnerabilities*. Retrieved April 1, 2021, from https://www.prnewswire.com/news-releases/puresec-reveals-that-21-of-open-source-serverless-applications-have-critical-vulnerabilities-300624175.html

Ltd, P. (2018b). *PureSec reveals that 21% of open source serverless applications have critical vulnerabilities*. Retrieved March 3, 2021, from https://www.prnewswire.com/news-releases/puresec-reveals-that-21-of-open-source-serverless-applications-have-critical-vulnerabilities-300624175.html

Lu, J. (2019). *Assessing the cost, legal fallout of Capital One data breach*. Retrieved from https://www.researchgate.net/profile/Jack-Lu-11/publication/335210159_Assessing_The_Cost_Legal_Fallout_Of_Capital_One_Data_Breach/links/5d58348992851cb74c74965c/Assessing-The-Cost-Legal-Fallout-Of-Capital-One-Data-Breach.pdf

Maissen, P., Felber, P., Kropf, P., & Schiavoni, V. (2020). FaaSdom: A benchmark suite for serverless computing. *Proceedings of the 14th ACM International Conference on*

Distributed and Event-Based Systems, pp. 73–84. https://doi.org/10.1145/3401025.3401738

Manor, E. (2018, July 24). *Bringing the best of serverless to you*. Google Cloud Platform Blog. Retrieved September 30, 2021, from https://cloudplatform.googleblog.com/2018/07/bringing-the-best-of-serverless-to-you.html

Many-faced threats to serverless security. (2017, August 14). Theburningmonk.com. Retrieved October 19, 2021, from https://theburningmonk.com/2017/08/many-faced-threats-to-serverless-security/

Mardesich, J. (2014, April 15). *Developers, check your Amazon bills for bitcoin miners*. Read Write. https://readwrite.com/2014/04/15/amazon-web-services-hack-bitcoin-miners-github/

McCumskey, G. (Oct 23, 2019). *Creating, monitoring, and testing cron jobs on AWS*. Retrieved January 29, 2022, from http://serverless.com//blog/cron-jobs-on-aws

Midwayjs. (2018). *Midway-a node.js serverless framework for front-end/full-stack developers. build the application for next decade. works on AWS, Alibaba Cloud, Tencent cloud and traditional VM/container. Super Easy integrate with react and Vue. - (midway).* Open-Source Libs. Retrieved June 19, 2022, from https://opensourcelibs.com/lib/midway

Mishra, A. (2022, June 19). Open-source software conference. O'Reilly Media-Technology and Business Training. Retrieved June 19, 2022, from https://www.oreilly.com/conferences/oscon.html

Mittal, A. (2019, February 7). *How to build a serverless backend with AWS Lambda*. Retrieved January 29, 2022, from https://pusher.com/tutorials/serverless-backend-aws-lambda/#defining-the-first-rest-api-route

Mohanty, S. K., Premsankar, G., & di Francesco, M. (2018). An evaluation of open source serverless computing frameworks. *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom),* pp. 115–120. https://doi.org/10.1109/CloudCom2018.2018.00033

Naor, G. (2018). *Six security considerations for serverless environments*. The New Stack. Retrieved from https://thenewstack.io/six-security-considerations-for-serverless-environments/

Nayak, S. (2020, October 19). *How to build a serverless application using AWS chalice*. Medium. https://towardsdatascience.com/how-to-build-a-serverless-application-using-aws-chalice-91024416d84f

Nick. (2021, March 14). *6 development tools for serverless applications.* The Iron.io Blog. Retrieved June 19, 2022, from https://blog.iron.io/6-development-tools-for-serverless-applications/

Node-lambda. (2021). Npm. Retrieved October 12, 2021, from https://www.npmjs.com/package/node-lambda

Okteto Team. (2020, July 2). *Serverless development with Kubeless and Okteto*. https://okteto.com/blog/serverless-development-with-kubeless/

OneLogin: Breach exposed ability to decrypt data. (2017, June 1). Krebs on Security. Retrieved June 16, 2022, from https://krebsonsecurity.com/2017/06/onelogin-breach-exposed-ability-to-decrypt-data/

Openfaas 8.0.4 · helm/openfaas. (2021). Artifact Hub. Retrieved October 7, 2021, from https://artifacthub.io/packages/helm/openfaas/openfaas

OWASP-Top-10-serverless-interpretation-en.pdf. (2017). Retrieved January 17, 2022, from https://owasp.org/www-pdf-archive/OWASP-Top-10-Serverless-Interpretation-en.pdf

Pauli, D. (2015). *Dev put AWS keys on Github. Then BAD THINGS happened*. Retrieved October 19, 2021, from https://www.theregister.com/2015/01/06/dev_blunder_shows_github_crawling_with_keyslurping_bots/

Perris, R. (2021, April 8). *Serverless security with automated API security testing*. Stack Hawk. Retrieved June 16, 2022, from https://www.stackhawk.com/blog/serverless-security-api-testing/

Persson, P., & Angelsmark, O. (2017). *Kappa: Serverless IoT deployment*, pp. 16–21. https://doi.org/10.1145/3154847.3154853

Rackerlabs. (2020). *Rackerlabs/lambda-uploader—Githubmemory*. Retrieved October 14, 2021, from https://githubmemory.com/repo/rackerlabs/lambda-uploader

Rahman, M. M., & Hasibul, H. M. (2019). Serverless architecture for big data analytics. *2019 Global Conference for Advancement in Technology (GCAT)*, pp. 1–5. https://doi.org/10.1109/GCAT47503.2019.8978443

Rehemägi, T. (2021a, July 24). Can we solve serverless cold starts? Dashbird. Retrieved June 19, 2022, from https://dashbird.io/blog/can-we-solve-serverless-cold-starts/

Rehemägi, T. (2021b, November 16). *How to measure and improve your serverless application's health*. Medium. https://towardsaws.com/how-to-measure-and-improve-your-serverless-applications-health-ae46f6a71138

Rehemägi, T. (2021c, December 1). Serverless security hazards and trends to consider. Dashbird. Retrieved June 16, 2022, from https://dashbird.io/blog/serverless-security-hazards/

Releases · aliyun/fcli. (2020). GitHub. Retrieved October 14, 2021, from https://github.com/aliyun/fcli/releases

Retter, M. (2021, June 6). *History of serverless computing: Where it started*. Dashbird. Retrieved January 29, 2022, from https://dashbird.io/blog/origin-of-serverless/

Ribenzaft, R. (2020, April 13). *Serverless open-source frameworks: OpenFaas, Knative, & More*. Cloud Native Computing Foundation. Retrieved June 19, 2022, from https://www.cncf.io/blog/2020/04/13/serverless-open-source-frameworks-openfaas-knative-more/

Rice, L. (2017). Could zombie toasters ddos my serverless deployment? *The New Stack.* Retrieved June 5, 2022, from https://thenewstack.io/zombie-toasters-eat-startup/

Roberts, M. (22 May 2018). *Serverless Architectures*. Martinfowler.Com. Retrieved January 23, 2022, from https://martinfowler.com/articles/serverless.html

Roberts, M., & Chapin, J. (2021a). *What is serverless?* O'Reilly Online Learning. Retrieved June 19, 2022, from https://www.oreilly.com/library/view/what-is-serverless/9781491984178/ch04.html

Roberts, M., & Chapin, J. (2021b). *What is serverless?* O'Reilly Online Learning. Retrieved June 19, 2022, from https://www.oreilly.com/library/view/what-is-serverless/9781491984178/ch04.html

Rojas, D. A. (2020, August 11). *Agnostic serverless functions using Kubeless*. QuintoAndar Tech Blog. https://medium.com/quintoandar-tech-blog/agnostic-serverless-functions-using-kubeless-3e0198c123eb

Sampe, J., Garcia-Lopez, P., Sanchez-Artigas, M., Vernik, G., Roca-Llaberia, P., & Arjona, A. (2021). Toward multicloud access transparency in serverless computing. *IEEE Software, 38*(1), 68–74. https://doi.org/10.1109/MS.2020.3029994

Sciabarra, M. (2021). *Learning Apache open whisk [book]-O'Reilly online learning.* O'Reilly. Retrieved June 20, 2022, from https://www.oreilly.com/library/view/learning-apache-openwhisk/9781492046158/ch01.html

Security risks arising from serverless technology. (2021, January 21). Haltdos.com. Retrieved January 17, 2022, from https://www.haltdos.com/blog/security-risks-arising-from-serverless-technology

Sengupta, S. (Jan 21, 2021). *Guide to debugging serverless applications.* LinkedIn. Retrieved January 23, 2022, from https://www.linkedin.com/pulse/guide-debugging-serverless-applications-sudip-sengupta/

Serverless advantages and use cases. (2022). Dashbird. Retrieved January 29, 2022, from https://dashbird.io/knowledge-base/basic-concepts/serverless-advantages-and-use-cases/

Serverless CI/CD: Built for serverless applications. (2022). Retrieved January 29, 2022, from http://serverless.com//ci-cd

Serverless cloud function Tencent cloud. (2021). Retrieved October 8, 2021, from https://intl.cloud.tencent.com/product/scf

Serverless computing–Amazon Web Services. (2022). Amazon Web Services, Inc. Retrieved January 30, 2022, from https://aws.amazon.com/serverless/

Serverless debugging guide. (2022). Lumigo. Retrieved January 23, 2022, from https://lumigo.io/debugging-aws-lambda-serverless-applications/

Serverless for Kubernetes with fission functions as a service. (2021). Platform9. Retrieved October 12, 2021, from https://platform9.com/fission/

Serverless framework—AWS lambda guide—Introduction. (2021). Retrieved September 19, 2021, from https://serverless.com/framework/docs/providers/aws/guide/intro

Serverless framework—Azure functions guide—Testing. (2021). Retrieved September 19, 2021, from https://serverless.com/framework/docs/providers/azure/guide/testing

Serverless infrastructure providers. (2020). Serverless. Retrieved February 23, 2021, from https://serverless.com/framework/docs/providers/

Serverless security for serverless containers and functions. (2022, January 3). Aqua. Retrieved January 17, 2022, from https://www.aquasec.com/products/serverless-container-functions/

Serverless Web IDE Tencent cloud. (2021, April 12). Retrieved October 15, 2021, from https://intl.cloud.tencent.com/document/product/583/39962

Shafiei, H., Khonsari, A., & Mousavi, P. (2019). *Serverless computing: A survey of opportunities, challenges and applications.* ArXiv:1911.01296 [Cs]. http://arxiv.org/abs/1911.01296

Shankar, V., Krauth, K., Pu, Q., Jonas, E., Venkataraman, S., Stoica, I., … & Ragan-Kelley, J. (2018). *Numpywren: Serverless linear algebra*. ArXiv:1810.09679 [Cs]. http://arxiv.org/abs/1810.09679

Sheridan, K. (2019, July 30). *Capital one breach affects 100m US citizens, 6m Canadians.* Dark Reading. Retrieved June 5, 2022, from https://www.darkreading.com/cloud/capital-one-breach-affects-100m-us-citizens-6m-canadians

Shilkov, M. (2021, January 5). *Comparison of cold starts in serverless functions across AWS, Azure, and GCP*. Retrieved January 23, 2022, from https://mikhail.io/serverless/coldstarts/big3/applications/

Shivang. (2019, January 15). *What is Apache OpenWhisk? Why use it? Everything you should know about it*. Scaleyourapp.Com. https://www.scaleyourapp.com/what-is-apache-openwhisk-why-use-it-everything-you-should-know-about-it/

Singh, G. (2012, February 3). *Kubeless—Kubernetes native serverless framework*. Retrieved June 19, 2022, from https://www.xenonstack.com/insights/kubeless

Sitapara, J. (2018, October 19). *Real-time data processing with serverless computing.* DATAVERSITY. https://dev.dataversity.net/real-time-data-processing-serverless-computing/

SiteSpirit. (2019, February 13). https://www.ibm.com/case-studies/sitespirit

Sollow, H. (2020, July 10). *Top 4 reasons why serverless is secure.* Check Point Software. Retrieved June 16, 2022, from https://blog.checkpoint.com/2020/07/13/top-4-reasons-why-serverless-is-secure/

Dark Reading Staff. (2019, October 24). *Eight-hour ddos attack struck AWS customers.* Dark Reading. Retrieved June 19, 2022, from https://www.darkreading.com/cloud/eight-hour-ddos-attack-struck-aws-customers/d/d-id/1336165

Taibi, D., Spillner, J., & Wawruch, K. (2021). Serverless computing-where are we now, and where are we heading? *IEEE Software*, *38*(1), 25–31. https://doi.org/10.1109/MS.2020.3028708

Tal, L., & Podjarny, G. (2019a, May 31). *10 serverless security best practices.* Snyk. Retrieved June 19, 2022, from https://snyk.io/blog/10-serverless-security-best-practices/

Tal, L., & Podjarny, G. (2019b, May 31). *10 serverless security best practices.* Snyk. Retrieved June 19, 2022, from https://snyk.io/blog/10-serverless-security-best-practices/

Tanasa, W. (2019, March 21). *Mitigating serverless lock-in fears*. Thought works. Retrieved June 19, 2022, from https://www.thoughtworks.com/en-us/insights/blog/mitigating-serverless-lock-fears

Taylor, T. (2018a, August 24). *Top 8 tools to use when working with serverless computing.* TechGenix. Retrieved June 16, 2022, from https://techgenix.com/serverless-computing-tools/

Taylor, T. (2018b, August 24). *Top 8 tools to use when working with serverless computing.* TechGenix. Retrieved June 16, 2022, from https://techgenix.com/serverless-computing-tools/

The 50 most preferred open-source serverless tools. (2021). Thechief.Io. Retrieved October 15, 2021, from https://thechief.io/c/editorial/the-50-most-preferred-open-source-serverless-tools/

The ten most critical risks for serverless applications v1.0. (2022). PureSec. https://github.com/puresec/sas-top-10 (Original work published 2018)

Timmerman, G. (2020, October 26). *Serverless functions in any language.* Google Cloud - Community. https://medium.com/google-cloud/serverless-functions-in-any-language-b44401c40859

Tozzi, C. (2021, March 18). *Microservices vs. serverless architecture*. Sumo Logic. Retrieved January 29, 2022, from https://www.sumologic.com/blog/microservices-vs-serverless-architecture/

Tseggai, N. (2018, July 12). *Disaster recovery in a serverless world - part 1.* Stackery RSS. Retrieved June 19, 2022, from https://www.stackery.io/blog/disaster-recovery-in-a-serverless-world-1-of-2/

Use fcli for the first time—Legacy Alibaba Cloud Documentation Center. (2020). Retrieved October 14, 2021, from https://partners-intl.aliyun.com/help/doc-detail/52995.htm

User, S. (2021). *Security of serverless applications: Key challenges and best practices to overcome them.* Apriorit. Retrieved March 16, 2021, from https://www.apriorit.com/dev-blog/657-web-security-of-serverless-applications

Vijayan, J. (2021). *Serverless vendor lock-in: Should you be worried?* TechBeacon. Retrieved October 19, 2021, from https://techbeacon.com/enterprise-it/serverless-vendor-lock-should-you-be-worried

VISH. (2020, March 26). *DDoS defence—AWS serverless architecture*. Medium. https://medium.com/@IAMVISH/ddos-defence-aws-serverless-architecture-e3027f85278e

Wang, A., Chang, S., Tian, H., Wang, H., Yang, H., Li, H., … & Cheng, Y. (2021). *FAASNET: Scalable and fast provisioning of custom serverless container runtimes at Alibaba Cloud Function Compute*. p. 16.

Wang, H., Niu, D., & Li, B. (2019). Distributed machine learning with a serverless architecture. *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 1288–1296. https://doi.org/10.1109/INFOCOM.2019.8737391

Web Application Firewall (WAF) solutions. (2022, February 13). Reblaze. Retrieved January 17, 2022, from https://www.reblaze.com/product/web-application-firewall/

Welcome to meteoroid documentation. (2021). Fiware. Retrieved June 19, 2022, from https://fiware-meteoroid.readthedocs.io/en/latest/

Werner, S., Girke, R., & Kuhlenkamp, J. (2020). An evaluation of serverless data processing frameworks. *Proceedings of the 2020 Sixth International Workshop on Serverless Computing*, pp. 19–24. https://doi.org/10.1145/3429880.3430095

What is a DDoS attack? (2020). Digital Attack Map Retrieved April 1, 2021, from https://www.digitalattackmap.com/understanding-ddos/

What is function compute. (2018). Alibaba Cloud. Retrieved October 3, 2021, from https://partners-intl.aliyun.com/help/doc-detail/52895.htm

What Is serverless computing? What are the features of serverless? (2021). Alibaba Cloud. Retrieved October 1, 2021, from https://www.alibabacloud.com/knowledge/what-is-serverless

What is the CIA triad? (2021, May 5). Forcepoint. Retrieved June 16, 2022, from https://www.forcepoint.com/cyber-edu/cia-triad

Yagolnitze, Y. (2020, June 3). *Preventing injection attacks on serverless applications*. Reblaze.com. Retrieved June 19, 2022, from https://www.reblaze.com/blog/preventing-injection-attacks-on-serverless-

Yegulalp, S. (2019, August 28). *7 open source tools that make AWS Lambda better*. InfoWorld. https://www.infoworld.com/article/3434007/7-open-source-tools-that-make-aws-lambda-better.html

Yusuf, S. (2019, May 7). *Using IoT with serverless to tackle global issues*. Thundra. https://medium.com/thundra/using-iot-with-serverless-to-tackle-global-issues-9695a40d5d8d

Zhang, T., Xie, D., Li, F., & Stutsman, R. (2019). Narrowing the gap between serverless and its state with storage functions. *Proceedings of the ACM Symposium on Cloud Computing*, pp. 1–12. https://doi.org/10.1145/3357223.3362723