Serverless Computing Architecture Security and Quality Analysis for Back-end Development

Clark Jason Ngo
School of Technology & Computing
City University of Seattle
Seattle, USA
clarkngo@cityuniversity.edu

Peng Wang
School of Technology & Computing
City University of Seattle
Seattle, USA
wangkevin@cityuniversity.edu

Tuan Khai Tran
School of Technology & Computing
City University of Seattle
Seattle, USA
t.k.tran@cityuniversity.edu

Sam Chung
School of Technology & Computing
City University of Seattle
Seattle, USA
chungsam@cityu.edu

***Abstract*—The purpose of this paper is to propose how to improve both quality and security for the back-end of a modern software system through adapting to the serverless computing architecture. For this purpose, this paper will conduct the following three steps: 1) Show a complete back-end architecture using three serverless computing such as Amazon Web Service (AWS), Microsoft Azure (Azure), and Google Cloud Platform (GCP). 2) Analyze each component's security and quality of each serverless computing provider and compare it to show similarities and differences. 3) Describe how using a cloud service improves the quality and security of a system.**

Keywords*—Serverless Development, Amazon Web Services, Microsoft Azure, Google Cloud Platform, Security Design, Quality Improvement

I. Introduction

The on-premise solution allows complete control of the security level. However, on-premise tends to fall behind in its security because it is rigid, time-consuming, and expensive. Moreover, it is difficult to maintain and upgrade. It is not able to quickly adapt to the ever-changing security best practices and compliance to combat cyber-attacks and meet the minimum regulatory requirements.

An international organization using the traditional on-premise solution requires them to provision, operate, and maintain multiple office spaces, data centers, developers, system administrators in multiple locations, and the application on their own. See Figure 1 for the on-premise setup.

Fig. 1. Traditional On-Premise Solution.

How can we make the process better? Cloud computing. With cloud computing, organizations can tap into a third party's data center and use their infrastructure. The organization still has to provision and operate these services with less overhead expenses such as developers and system administrators. Instead of capital expense for servers and data centers, the organization would incur operating costs for renting their infrastructure from third-parties offering these services. Figure 2 shows the organization can focus on housing developers and system administrators in their headquarters.

Fig. 2. Cloud Computing Solution with data centers shown in a different color to signify non-ownership of the infrastructure of the organization.

How can we make the process better? Enter serverless computing. No need to provision, operate, and maintain data centers. With that, serverless computing solution has less code base and nothing to manage hence requiring fewer developers and system administrators than the cloud computing solution. Figure 3 shows the serverless computing.

Fig. 3. Serverless Computing Solution with Less Code Base

Serverless computing is an event-driven model or code execution where infrastructure is abstracted from the developer or end-user [1]. These applications using

979-8-6425-4896-7/20/$26.00 ©2020 CISSE

serverless computing are run in the cloud. However, these serverless applications that are managed by third parties are also run on physical servers. Sounds familiar? Cloud is just someone else's computer. So why use a serverless computing model then if it's just another physical server? Adopting a cloud-based service can downscale, bringing the costs to almost 1/10 compared to maintaining our infrastructure [2]. It provides ease of provisioning, operating, and maintaining. Cloud providers are compliant with most industry standard and regulatory requirements. Cloud helps companies focus more on business logic rather than spend time and resources for provisioning, maintaining, and operating resources [3].

Serverless applications provide high availability, fault-tolerant, scalability, and elasticity. With all the benefits serverless architecture brings, it also has some limitations. These limitations are controlling limit on infrastructure, locked-in with a vendor, and the impact of cold start. Moreover, moving the cloud would directly expose our servers to the internet [3].

In managing our cloud services, the principle of do-not-trust-anyone should be applied. This paper will show how a serverless architecture can impact both security and quality for a web application. The analysis will compare the cloud services of Amazon Web Services (AWS), Microsoft Azure (Azure), and Google Cloud Platform (GCP).

II. The Architecture for Web Apps

A simple case of web application architecture. A client can retrieve a static content in a Storage (File) Server. The client can make an API call to a Compute Server (function) that is authenticated through an authentication service. The function can then query the Database Server. The Logging and Monitoring Service will collect metrics and logs from all the services. Figure 4 demonstrates the typical flow of requests in a web application.

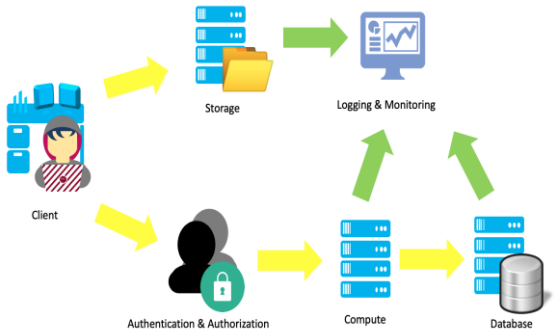

Fig. 4. Web Application Architecture.

AWS offers various serverless services that can allow engineers to make a fully serverless back-end application. Potential architectures for web and mobile applications are shown in Figure 5 for AWS and Figure 6 for Azure. GCP also has similar services, which can help to make a similar architecture. The equivalent service names can be found in Table I.

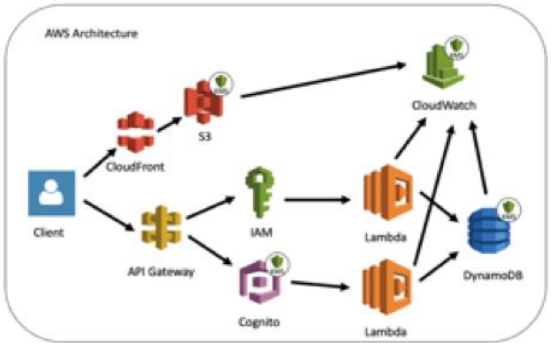

Fig. 5. AWS Serverless Architecture.

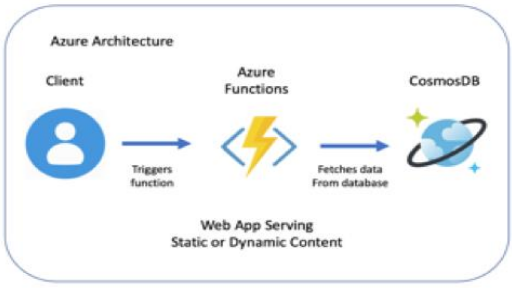

Fig. 6. Azure Serverless Architecture.

TABLE I. A Comparison of Service Name of AWS, Azure, and GCP

Service Type	Service Names		
	AWS	***GCP***	***Azure***
Storage	S3	Cloud Storage	Azure Blob
Network Gateway	API Gateway	Cloud VPN	VPN Gateway
Authentication & Authorization	IAM	Cloud IAM	Active Directory
Compute	Lambda	Cloud Functions	Azure Functions
Database	DynamoDB	Cloud SQL	Cosmos DB
Logging Monitoring	CloudWatch	Google Stackdriver	Azure Monitor

III. Security and Quality Analysis

Each component in the architecture, as shown in Figure 4, will be analyzed from security and quality perspectives to see whether using serverless technologies can increase both security and quality.

A. Storage Service

In the AWS architecture, the Simple Storage Service (S3) is chosen for the storage service that will store all static content such as HTML files, JavaScript files, images,

Lambda function codes, and so forth. By using S3, four methods can be used for conducting encryption at rest to protect users' data [2]. The four methods are: 1) Data at rest inside the volume, 2) All data moving between the volume and the instance, 3) All snapshots created from the volume, and 4) All volumes created from those snapshots. The CloudFront will help S3 to set up the Secure Sockets Layer (SSL) to apply the encryption in transit, which can protect data between clients and the cloud [4]. From the quality perspective, according to AWS [5], S3 can automatically give users at least 3,500 write with 5,500 read requests per second per prefix in a bucket. The document also mentioned that there are no limits to the number of prefixes in a bucket, which means S3 gives users the ability to conduct unlimited read and write concurrent requests by adding more prefixes for the same file. It is extremely hard and costly to do the same with a non-serverless solution due to the hardware or provision needs to be prepared ahead.

In the Google Cloud architecture, Cloud Storage is a unified object storage to store live or archive data [6]. Data is encrypted at rest and located on the server-side with Cloud Storage. At the storage system layer, Google uses the Advanced Encryption Standard (AES) for encrypting data at rest. At the storage device layer, data is encrypted with AES 128 for hard disk drives and AES256 for solid-state drives. This encryption at the storage device layer is done with the device-level key. At backups, files are encrypted with Data Encrypted Key (DEK) [7]. According to Google Cloud [8], best practices for securing a Google Cloud Storage are as follows: 1) Have distinct credentials per user, never share the credentials, and securely store them. 2) Always use Transport Layer Security (TLS) such as HTTPS to transmit data. 3) Use Hypertext Transfer Protocol Secure (HTTPS) library that validates server certificates to mitigate the risk of man-in-the-middle-attack. 4) Revoke authentication credentials if an application does not need access anymore. 5) Create a bucket name with names difficult to guess and do not add sensitive information such as our account or Application Programming Interface (API) secret keys as part of the bucket name. In terms of quality, Cloud Storage can provide 1000 writes per second and unlimited reads. Scaling is provided for both writes and reads [9]. Availability is improved through scheduled deletion instead of immediate. This scheduled deletion allows recovery from accidental deletions or deletion from internal bugs or processes [10].

In the Microsoft Azure architecture, Azure also provides the number of services for data storage, including Azure Blob, Azure Files, Azure queue, and Azure tables. Azure storage offers data stores for different objects such as a data file repository, an archive for messaging channels, and NoSQL datastore. Azure Blob is probably equivalent to Amazon S3 as an object storage solution that can be used to store all types of unstructured data [11]. With the use of Representational State Transfer (REST)-ful, Blob store manages to reduce, and Azure platform ensures that objects will be stored and available on multiple storage copies, which enhances Availability in Confidentiality, Integrity, and Availability (CIA) triad. By replicating data, it also helps to protect user's data in both planned and unplanned events [12]. Microsoft Azure storage services are also known for its collection of security capabilities. Role-Based Access Control (RBAC) allows users to secure their storage account by restricting accesses following the least privilege principle. Data transition across networks will also be secured by encrypting with HTTPS, SMB 3.0, and client-side encryption. They also provide Azure Disk Encryption for virtual machines and data disks, Advanced Threat Protection for monitor Storage logs, and check for any suspicious requests to Block Blob storage.

All three service names provide data encryption at rest and data encryption in transit in the aspect of security. They also provide high write, read, and durability in the aspect of quality. The high durability delivered by cloud service providers is hard to achieve by an organization as it requires huge expenses for infrastructure to build your own. See Table II for a summary of storage service.

TABLE II. STORAGE SERVICE / FILE SERVER

Analysis	Service Names		
	AWS	*GCP*	*Azure*
Security Assurance			
Data encrypted at rest	Yes	Yes	Yes
Data encrypted in transit	Yes	Yes	Yes
Quality Assurance			
Write	High	High	High
Read	High	High	High
Durability	High	High	High

B. Authentication and Authorization Service

Three services that include API Gateway, Identity and Access Management (IAM), and Cognito are used in the AWS architecture to achieve the authentication and authorization functionality. A user's request will reach the endpoint that offers by API Gateway and then passes to either IAM for providing accesses with their infrastructure or Cognito for providing accesses to outside users. Cognito supports both encryptions at rest and in transit when it is Health Insurance Portability and Accountability Act (HIPAA) eligible and Payment Card Industry Data Security Standard (PCI DSS), Service Organization Control (SOC), ISO/EIC 27001, International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC) 27017, ISO/IEC 27018, and ISO 9001 compliant [2]. From the quality perspective, Cognito offers the capabilities of fully managing the user lifecycle and automatically verifying identities from API Gateway without customized implementation required out of the box. It can improve the system quality a great deal by avoiding to use an

incorrect or weak authentication implementation. API Gateway also offers the cache capability without requiring to change any code plus the ability to control the API usage for each of our clients for free.

GCP has Cloud IAM to allow administrators to have a platform to manage and control user and group access for cloud services [13]. IAM enforces policies to member identities and roles. Roles are derived from the collection of permissions. Figure 7 shows GCP's IAM. It adopts the principle of least privilege. IAM configurations are on a per-project basis. A project must exist first before adding users and roles. Cloud Endpoints manages our API's development, deployment, and monitoring [14]. Cloud Endpoints works in conjunction with Cloud IAM to grant and revoke API access. Google Cloud is compliant with ISO 27001, ISO 27017, ISO 27018, SOC 1/2/3, PCI DSS, and Cloud Security Assurance (CSA) Security Trust Assurance and Risk (STAR) [6].

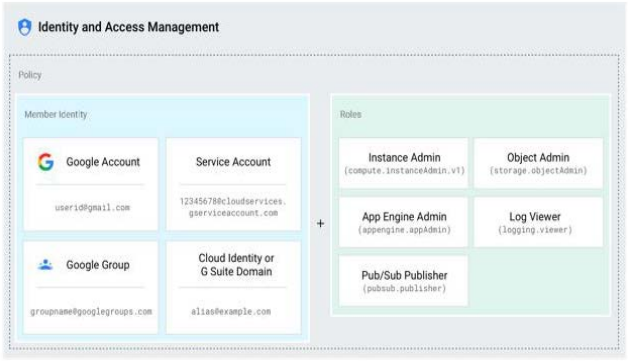

Fig. 7. GCP Identity and Access Management [6].

In GCP, the system is improved with encryption of inter-service communication, which provides automatic encryption for infrastructure Remote Procedure Call (RPC). End-user data access is equipped with automatic mutual encryption, see Figure 8 [10]. Insider risk is reduced through two-party approval implementation for specific actions and API debugging without exposure to sensitive information [10].

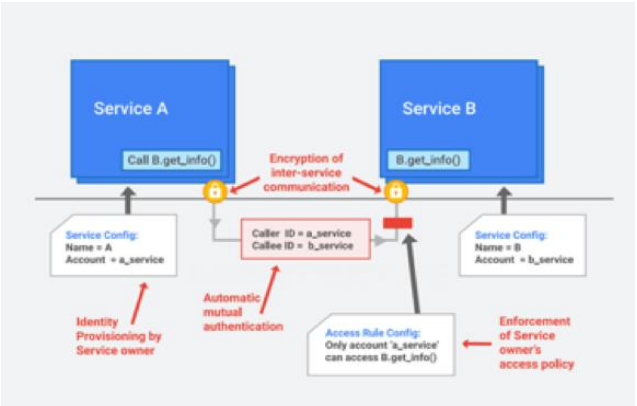

Fig. 8. GCP Service Identity, Mutual Authentication, and Encrypted Inter-Service [10].

All three service names provide IAM to access and control resources for free in the aspect of security. They also allow better authentication implementation in the aspect of quality. Organizations creating their authentication service run the risk of incorrect or weak authentication implementation as it is not the core of their service. See Table III for a summary of authentication and authorization service.

TABLE III. AUTHENTICATION AND AUTHORIZATION SERVICE

Analysis	Service Names		
	AWS	*GCP*	*Azure*
Security Assurance			
IAM to access and control resources	Yes	Yes	Yes
Quality Assurance			
Avoids incorrect or weak authentication implementation	Yes	Yes	Yes

C. *Computing Service*

The Lambda service is chosen for AWS architecture's computing component. Lambda itself does not help to improve the security directly since it is hiding behind API Gateway. However, the security level can be raised indirectly because the Lambda service will handle the OS level and runtime level patching automatically, which will reduce the risk of having vulnerabilities significantly. Additionally, it also helps in raising the quality of the system a great deal. By using Lambda, users can automatically scale in and out based on the real-time traffic without even thinking about the provision. It gives end users a much better user experience.

Cloud Function is a computing service provided by GCP that lets us run code in the cloud. It shines best when used for microservices as it adds agility by having small independent functionalities running instead that one tremendous service [15]. As of February 27, 2019, the quota for function calls is at 40,000,000 in all regions. Security is still in the hands of the developer. The developer should follow best practices such as passing a JSON Web Token (JWT) to our functions, attaching a role with least privileges, hard to guess function name, and logging and monitoring using Stackdriver to enhance security [16].

Similar to the Lambda and Cloud Function, Azure Function is also a service that allows users to run small pieces code or function on the cloud to solve a specific problem without bothering other components related to the application as well as the infrastructure to run this function. Azure Function was built based on a trigger and binding mechanisms. Triggers are used to make the functions run, which will rely on a specific event to activate our function. Binding is a way to declare used services to simplify the interaction with input and output data [17]. To optimize the performance, Azure Function can be integrated with some other Azure services such as Cosmos DB, Azure Event Hubs,

Azure Storage Blob. These services can be used to execute our functions through some provided templates, including CosmosDBTrigger, BlobTrigger, QueueTrigger, and EvenHubTrigger.

Azure Active Directory is a Microsoft Azure solution for IAM. The service is used to manage access to employees, partners, customers, and corporate assets. Users can synchronize their on-premise local infrastructure with Windows Azure Active Directory to provide single sign-on for users to access cloud applications [18]. Azure Active Directory also brings multi-authentication to the table as an additional defense to help a business protect sensitive information and applications. Some of the best practices that need to be considered when working with Azure Active Directory include enabling single sign-on, enforcing multi-authentication, applying role-based access control, centralizing identity management, and being aware of locations where resources are placed [19].

All three service names do not offer any security for the compute service. The developer is responsible for building secure codes. Compute service delivered by cloud providers are exceptional in quality aspect as it enables organizations to scale-in and out based on real-time traffic. See Table IV for a summary on compute service.

TABLE IV. COMPUTE SERVICE / BACKEND SERVER

Analysis	Service Names		
	AWS	*GCP*	*Azure*
Security Assurance			
Any security?	No	No	No
Quality Assurance			
Scale-in and out based on real-time traffic	Yes	Yes	Yes

D. Database Service

Several serverless database options can be chosen for the AWS architecture. DynamoDB is one of them that can deliver single-digit millisecond performance at any scale [20]. The official document [20] also mentioned that "DynamoDB can handle more than 10 trillion requests per day and support peaks of more than 20 million requests per second." It makes DynamoDB is an excellent candidate for any mission-critical workload. DynamoDB encrypts all customers' data at rest by default, which is an excellent security feature that users can get for free [21]. According to the official document [21], using DynamoDB also can help to improve the system quality through different features it has. First of all, it has an auto-scaling and on-demand mode that allows users to scale up and down based on real-time traffic ultimately. Secondly, DynamoDB can be used with DynamoDB Accelerator to improve the read performance by up to 10 times. Moreover, the DynamoDB Streaming feature could help users to take advantage of real-time data processing. Additionally, the built-in point-in-time recovery feature gives users the ability to restore a table to any point of time up to the second during the previous 35 days.

GCP offers various database services. Cloud SQL gives us high performance, scalable, and manageable PostgreSQL and MySQL databases in the cloud [22]. Cloud SQL has two levels of access controls. First, instance-level access authorizes access from an applicant, client, or other GCP services to our Cloud SQL instance. Second, database access uses MySQL Access Privilege System to manage MySQL users in our Cloud SQL instance [23]. Cloud SQL access and permissions can also be configured using Cloud IAM [24]. High availability is an option for a Cloud SQL project that works by creating a replica of an instance through semi-asynchronous replication in different zones [25].

Some of the more popular products when it comes to database services for Microsoft Azure are SQL Database and Cosmos DB. Azure Cosmos DB is known for its support on the multi-model and the ability to globally distribute for any scale. It is capable of replicating all the data in the database to a global scope with more than 30 Microsoft Azure data centers located throughout the world. With the ability to replicate such global scope, application connected to Cosmos DB will have very low latency with 99% data processing, which will be guaranteed at less than 10ms for both read and write [26]. In terms of security, Azure Cosmos DB uses master keys and resource tokens to perform authentication and provide appropriate access to its data and resources. It also adds to the database another layer of protection in case of crashes based on the "failover" mechanism. When a problem such as power outage on all the datacenter or a national, multinational scale incident, Cosmos DB can automatically handle the situation [26]. Cosmos DB is certified for various standards, such as CSA START and ISO 20000-1:2011 [26]. All the data stored is also well encrypted.

All three service names provide data encryption at rest and data encryption in transit in the aspect of security. They all offer unlimited throughput with different metrics and terminologies. AWS uses Write Capacity Units (WCU) and Read Capacity Units (RCU). GCP uses Read Units (RUs). Azure uses Query Per Second (QPS). Recovery differs on the entry-level for all three service names, and all are upgradable. See Table V for a summary of database service.

TABLE V. DATABASE SERVICE / DATABASE SERVER

Analysis	Service Names		
	AWS	*GCP*	*Azure*
Security Assurance			
Data encrypted at rest	Yes	Yes	Yes
Data encrypted in transit	Yes	Yes	Yes

Analysis	Service Names		
	AWS	*GCP*	*Azure*
Quality Assurance			
Throughput	Unlimited	Unlimited RUs	Unlimited QPS
Recovery	Up to 35 days	Up to 30 days	Up to 90 days

E. Logging and Monitor Service

The CloudWatch is a service that is integrated with almost all other AWS services to offer logging and motoring capability. It is also used for this AWS serverless architecture. The CloudWatch logs are PCI and FedRamp compliant and also provide encrypted at rest as well as encrypted in transit as its standard security features [27]. The system quality also can be raised by using CloudWatch. According to the official document [27], the CloudWatch offers an easy way to collect and store logs, collect built-in metrics, push custom metrics, and create dashboards. Most importantly, the CloudWatch allows users to create alarms based on some specific metrics, which can be used to trigger other AWS services. It gives users a great ability to automate tasks.

Google Stackdriver is a logging and monitoring tool for GCP, AWS, on-premise, or hybrid. Stackdriver features are debugger, error reporting, rapid discovery, uptime monitoring, integrations, smart defaults, alerts, tracing, logging, dashboards, and profiling. It helps enable observability, working with multiple cloud services, create fast fixes, and gather full-stack insights [6].

Azure Monitor is the selection for logging and monitoring. There are two categories that data collected by the service fall into, which are metrics and logs. Metrics refer to a numerical value that interprets some data of a system. In contrast, logs contain records that organized into a different set of properties. Azure Monitor also has a responding mechanism to critical situations. The respondent can be an alert or sending emails to administrators who are responsible for solving the issue, or it can launch an automated procedure and attempt to fix the issue [28]. Another feature that Azure Monitor can provide is to visualize monitoring data. Collected Data and elements can be added into their dashboard, which allows combining both metrics and logs into charts and diagrams. Thus, it can leverage other Azure services for publishing to different audiences [28].

All three service names provide data encryption at rest and data encryption in transit in the aspect of security. They all offer rapid discovery, uptime monitoring, and create fast fixes. Basic logging and monitoring service are usually free from the cloud service provider whose services you are using. This service can drive down the cost of monitoring instances and avoid the headache of integrations with other third-parties. See Table VI for a summary of the logging and monitoring service.

TABLE VI. LOGGING AND MONITORING SERVICE

Analysis	Service Names		
	AWS	*GCP*	*Azure*
Security Assurance			
Data encrypted at rest	Yes	Yes	Yes
Data encrypted in transit	Yes	Yes	Yes
Quality Assurance			
Rapid discovery	Yes	Yes	Yes
Uptime monitoring	Yes	Yes	Yes
Create fast fixes	Yes	Yes	Yes

IV. SUMMARY OF SECURITY FOR CLOUD SERVICES PROVIDERS

AWS, GCP, and Azure all support encrypting data at rest and in transit that uses AES-256 encryption, which is the industry standard and a secure block cipher. IAM for all three cloud providers addresses the required access to security and resource control. However, the implementation of authorization for AWS, GCP, and Azure differs. In AWS, we can immediately create users, groups, policies, and roles. In GCP, we need to create a project or choose an existing project before we can assign users, roles, etc. In Azure, it uses the subscription scope that has an owner, contributor, or reader, which grants different levels of access to resource groups depending on the role. The resource group would also have an owner, contributor, or reader. The concept of resource groups is perfect for isolating resources, as shown in Figure 9 [29].

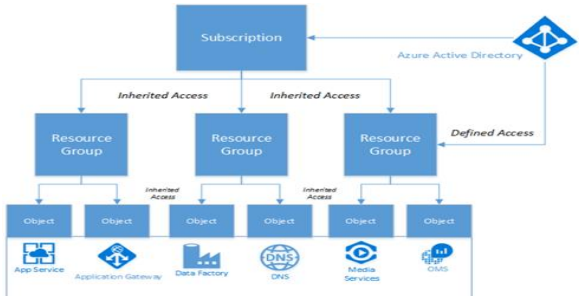

Fig. 9. Azure Active Directory Subscription and Resource Group [30].

V. FINDINGS

A. In Terms of Improving Security

Serverless architecture improves all aspects of security, such as authentication, authorization, encryption, confidentiality, integrity, and availability. Most providers

come with built-in monitoring and logging [1]. It delivers ease of control, management that requires less technical knowledge to implement better security.

Moreover, cloud providers are having better security standards as they specialize in the business of securing resources and applications. It is important to note that while cloud vendors improve cloud security such as network or infrastructure security, cloud users must ensure the security of their APIs. See Table VII for API access types.

TABLE VII. API ACCESS TYPES

API access type	Explanation	Example
Public APIs	Public content	Your homepage index.html
Internal APIs	APIs only called by other your functions	Function to call the data from the database
Authenticated APIs	APIs only usable for users	Unregistered users are not allowed to call API to mitigate DDoS attacks from API call abuse
APIs for third parties	APIs shareable to others	Integration for other third-party applications

B. In Terms of Improving Quality

Improving quality improves security. Serverless architecture improves quality by enforcing a consistent configuration standard for instantiating resources. Tools for monitoring, logging, and reviewing are readily available to vulnerability checks. Less code to maintain means less technical debt and fewer vulnerabilities from the codebase. Availability is vastly improved through auto-scaling, durability, and redundancy.

On the downside, serverless architecture has the following considerations that are detrimental to quality if not managed properly. These considerations are ephemeral functions, cold start, limited database configuration, and no system-level access. See Table VIII for architectural considerations for serverless computing [1].

TABLE VIII. ARCHITECTURAL CONSIDERATIONS

Considerations	Explanation	Workaround
Ephemeral functions	Functions are deployed in a container that they persist for a certain period only	Don't use for large processing requirements
Cold start	Invoking function for the first time after it was inactive increases execution time	Keep the function hot (stay alive) before execution
Limited DB configuration	Limit to simultaneous connection in relational DBs	Use NoSQL
No system-level access	No support for reading attributes from configuration files or spilling over to in-memory cache to disk	Don't use for applications needing file system level or operating system level access

VI. SECURITY CONCERNS

Serverless architecture invites new security risks. Serverless functions receiving input from event sources and may include event messages injected with malicious data. The microservices-oriented design makes it difficult to build proper authentication due to the scale of serverless functions. The auto-scaling of serverless functions can lead to high financial cost if attacked with Denial-of-Service (DoS) [31].

VII. CONCLUSIONS

Serverless architecture drastically reduces security vulnerabilities from legacy code and resource configurations. AWS, GCP, and Azure all currently provide the minimum regulatory requirement and best practices. Choosing a cloud service provider would depend on the use case and extra security features on a particular service. A cloud provider might provide more insightful logging and review over the other or offer better availability that does not immediately shut down our resources after hitting the maximum instance or call made to a service.

Some concerns on serverless computing: 1) controlling limit on infrastructure as infrastructure is abstracted away from the developer, 2) Vendor lock-in makes it harder to integrate other services outside of the cloud provider's ecosystem, 3) Unable to use a monitoring service using a different third-party tool, (4) data injection on event message consumed, (5) authentication implementation fatigue, and (6) financial resource depletion due to DoS attacks.

It is important to follow industry guidelines to leverage the use of serverless architecture and improve overall security and quality of the application. See Table IX for industry guidelines on serverless architecture implementation.

TABLE IX. SERVERLESS ARCHITECTURE INDUSTRY GUIDING PRINCIPLES (DELOITTE, 2018)

Principles	Explanation	Principles
Develop simple purpose functions	Use single-purpose codes for more effortless deployment and test	Develop simple purpose functions

Principles	Explanation	Principles
Design push-based, event-driven patterns	Allow event chain to propagate without the need for user input	Design push-based, event-driven patterns
Create thicker and powerful frontends	Reduce function calls by executing more complex frond-end through rich client-side application framework	Create more abundant and powerful frontends
Incorporate appropriate security mechanism across the technology stack	Use of API Gateway layer and other mechanisms such as access controls, authentication, IAM, encryption, etc.	Incorporate appropriate security mechanism across the technology stack
Identify performance bottleneck	Measure performance of bottlenecks and identity functions slowing a service	Identify performance bottleneck

REFERENCES

[1] Deloitte. (2018). Google Cloud Security Retrieved from https://www2.deloitte.com/content/dam/Deloitte/tr/Documents/technology-media-telecommunications/Serverless%20Computing.pdf

[2] AWS. (2019a). AWS Compliance Programs. Retrieved from https://aws.amazon.com/compliance/programs

[3] Ayoub, D., & Wacker, R. (2012). Security: On-premise or in the cloud? Network World, 29(20), 18-19. Retrieved from http://proxy.cityu.edu/login?url=https://search-proquest-com.proxy.cityu.edu/docview/1265769745?accountid=1230

[4] AWS. (2019b). Amazon CloudFront custom SSL. Retrieved from https://aws.amazon.com/cloudfront/custom-ssl-domains

[5] AWS. (2019c). Request Rate and Performance Guidelines. Retrieved from https://docs.aws.amazon.com/AmazonS3/latest/dev/request-rate-perf-considerations.html

[6] Google Cloud. (2019a). Trust & Security. Retrieved from https://cloud.google.com/security/

[7] Google Cloud. (2019b). Encryption at Rest in Google Cloud Platform. Retrieved from https://cloud.google.com/security/encryption-at-rest/default-encryption/

[8] Google Cloud. (2019c). Best Practices for Google Cloud Storage. Retrieved from https://cloud.google.com/storage/docs/best-practices

[9] Google Cloud. (2019d). Stackdriver. Retrieved from https://cloud.google.com/stackdriver/

[10] Google Cloud. (2019e). Serverless Computing – Architectural Considerations & Principles. Retrieved from https://services.google.com/fh/files/misc/security_whitepapers_march2018.pdf

[11] Microsoft Azure. (2019a). Blob storage. Retrieved from https://azure.microsoft.com/en-us/services/storage/blobs/

[12] Microsoft Azure. (2019b). Storage redundancy. Retrieved from https://docs.microsoft.com/en-us/azure/storage/common/storage-redundancy

[13] Google Cloud. (2019f). Cloud Identity & Access Management. Retrieved from https://cloud.google.com/iam/

[14] Google Cloud. (2019g). Cloud Endpoints. Retrieved from https://cloud.google.com/endpoints/

[15] Google Cloud. (2019h). Google Cloud Functions. Retrieved from https://cloud.google.com/functions/

[16] Google Cloud. (2019i). Function Identity. Retrieved from https://cloud.google.com/functions/docs/securing/function-identity

[17] Microsoft Azure. (2019c). An Introduction of Azure Function. Retrieved from: https://docs.microsoft.com/en-us/azure/azure-functions/functions-overview

[18] Microsoft Corporation. (2018). Azure Functions and Serverless platform security. Retrieved from: https://azure.microsoft.com/mediahandler/files/resourcefiles/azure-functions-serverless-platform-security/Microsoft%20Serverless%20Platform.pdf

[19] Microsoft Azure. (2019d). Azure Identity Management and access control security best practices. Retrieved from: https://docs.microsoft.com/en-us/azure/security/azure-security-identity-management-best-practices

[20] AWS. (2019d). Amazon DynamoDB. Retrieved from https://aws.amazon.com/dynamodb

[21] AWS. (2019e). Amazon DynamoDB Features. Retrieved from https://aws.amazon.com/dynamodb/features

[22] Google Cloud. (2019j). Cloud SQL. Retrieved fromhttps://cloud.google.com/sql/

[23] Google Cloud. (2019k). Overview of High Availability Configuration. Retrieved from https://cloud.google.com/sql/docs/mysql/high-availability

[24] Google Cloud. (2019l). Project Access Control. Retrieved from https://cloud.google.com/storage/quotas

[25] Google Cloud. (2019m). Project Access Control. Retrieved from https://cloud.google.com/sql/docs/mysql/project-access-control

[26] Microsoft Azure. (2019e). Welcome to Microsoft Azure Cosmos DB. Retrieved from: https://docs.microsoft.com/en-us/azure/cosmos-db/introduction

[27] AWS. (2019f). Amazon CloudWatch Features. Retrieved from https://aws.amazon.com/cloudwatch/features

[28] Microsoft Azure. (2019f). Azure Monitor overview. Retrieved from: https://docs.microsoft.com/en-us/azure/azure-monitor/overview

[29] Power, V. (2018). IAM Roundup: AWS vs. Azure vs. GCP. Retrieved from https://www.twistlock.com/2018/09/25/iam-roundup-aws-vs-azure-vs-gcp/

[30] Sandbu, M. (2018). Microsoft Azure and Security Best Practices – Part 1 Identity. Retrieved from https://msandbu.org/microsoft-azure-and-security-best-pratices-part-1-identity/

[31] Segal, O. (2019). The 12 Worst Serverless Security Risks. Retrieved from https://www.darkreading.com/cloud/the-12-worst-serverless-security-risks/a/d-id/1334079