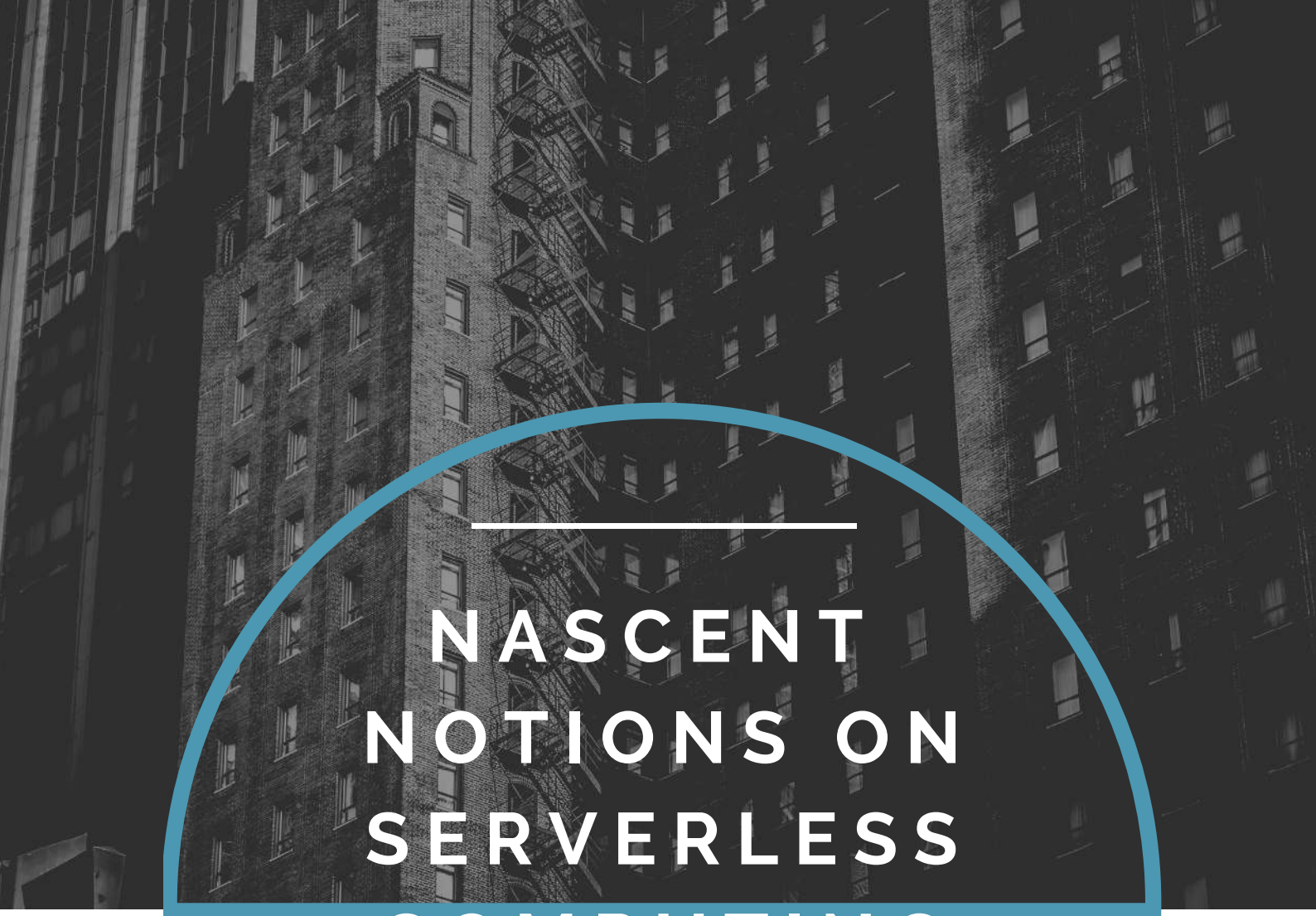


NASCENT NOTIONS ON SERVERLESS COMPUTING WHITEPAPER

PREPARED BY

Mr. Alex Xavier
Security Engineer
@briskinfosec Bint lab
www.briskinfosec.com.

PRESENTED BY

NCDRC
(National Cyber Defence Research Centre)
in collabration with
BINT Lab
www.ncdrc.res.in.

CONTENTS

1. INTRODUCTION

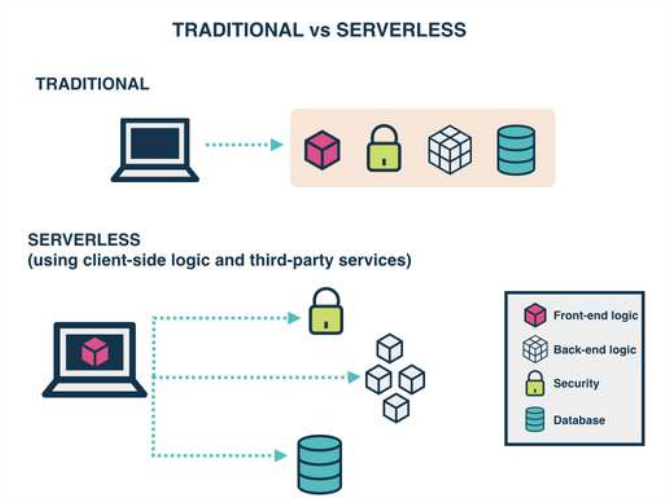
The most fascinating thing about technology is that, it never stops growing. It stuns people each and every time with its new inventions. One such feat of technology is Serverless computing. It's an emerging trend in the town and has been attaining a mass attention over a past couple of years. Both, the fresher and veteran enterprises in tech industry are adopting the Serverless computing feature with cloud service. It isn't just a hype but promises the organization with high productivity on a low-cost budget.

2. SERVERLESS COMPUTING

There is a common misconception about serverless, presuming as "No server", but it isn't. Well, servers are involved, but less used. Serverless is a Function as a Service (FaaS) software design pattern where applications are hosted by a third-party service, eliminating the need for server software and hardware management by the developer. Applications are broken up into individual functions that can be invoked and scaled individually.

3. SERVER BASED MODEL VS SERVERLESS MODEL

For a traditional application, it must first be developed and then deployed on the server with various needs like specific capacity planning, installing of server software and hardware, and also the constant monitoring of it. This might consume few weeks to months, and also involves a lot of Capital and Operational expenses. Whereas, Serverless application allows you to build and run applications and services by removing away the infrastructure requirements and relieving the developer from the operational complexities of running the application. In serverless, you have to provide only the code and functional logic for your application and then publish it to the cloud.



The rest is automatically taken care of by the cloud vendor. They will look after provisioning the server and deploy the function on it. As the demand for the function goes up, the cloud vendor will support with more servers and inactivate the service when the demand goes down. All of this will be transparent to the end user.

4. WHY TO USE SERVERLESS?

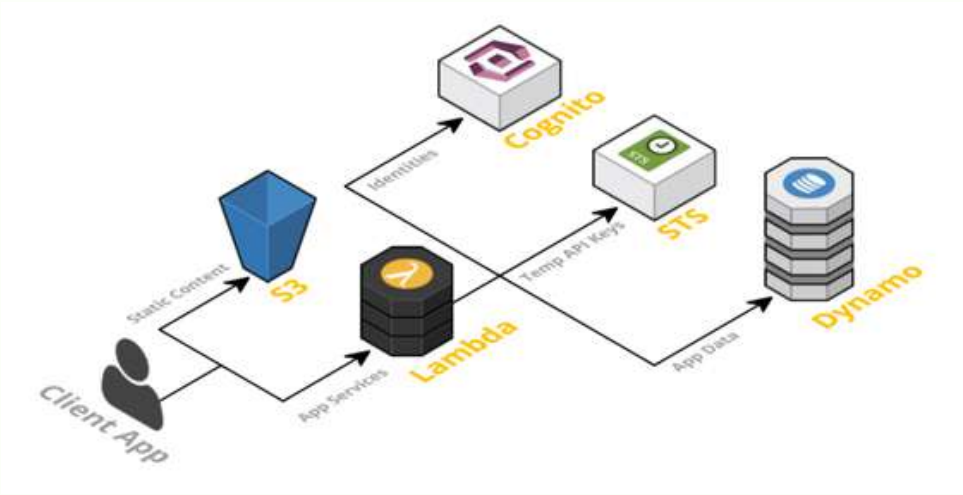
Serverless facilitates the developers to focus more on their application's core logic and resource utilization optimally for the application as well, building it with high availability, virtually at any scale without worrying about any servers. Serverless only fees for the actual amount of resource consumed by an application rather than on pre-purchased unit of capacity.

5. LEADING SERVERLESS PROVIDERS

Many cloud providers have heavily invested in serverless. Some of them are as follows:

6. THE SERVERLESS APPLICATION FUNCTION

A Serverless solution consists of a web server, Lambda functions (FaaS), Security Token Service (STS), user authentication and database.



- **Client Application**—Server-client application is the UI of your application that is rendered client side in Modern front-end JavaScript App. This enables us to use a simple static web server.

- **Web Server**—Server-client web server is the simple web server that is offered by the service provider (E.g. Amazon S3). All the static contents of the application are served from here (S3) (E.g. contents like HTML, CSS, JS files).

- **Lambda functions (FaaS)**—Acts as the FaaS and is also, the key enablers in serverless architecture (E.g. AWS Lambda, Microsoft Azure Function, Google Cloud Function). The Login and Data accessing of the application services will be built in the FaaS (Lambda functions). This function will READ and WRITE from the database as well, provide the JSON responses in reply.

- **Security Token Service (STS)**—STS will generate the temporary Credentials - API key and a secret key (E.g. AWS Credentials for the users of the application. The client application uses these temporary credentials to invoke AWS API).

- **User Authentication**—User Authentication provides the Identity Management System of the application. It enables the users to sign-up and sign-in to the application via mobile and web applications. (E.g. Here AWS Cognito identity service is integrated with AWS Lambda to authenticate users into the application. By doing so, it provides options for the user to sign-in and sign-up via social identity providers like Facebook, Twitter, or Amazon.

- **Database**—AWS DynamoDB provides a fully managed NoSQL database. DynamoDB is not essential for a serverless application but is used as an example here.

7. SERVERLESS FRAMEWORKS

Serverless platforms need a base to get executed. They also need a way to define and deploy Serverless codes on various cloud platforms or commercial services. Some of the Serverless frameworks are cited below:

- Serverless Framework (Javascript, Python, Golang)
- Apex (Javascript)
- ClaudiaJS (Javascript)
- Sparta (Golang)
- Gordon (Javascript)
- Zappa (Python)
- Up (Javascript, Python, Golang, Crystal)

8. WHAT CAN BE DONE WITH SERVERLESS?

- Static websites
- E-commerce platforms
- Image rich applications
- Chatbots
- IoT services
- Big Data applications
- Event-Driven systems

9. PROS OF SERVERLESS

- Simplifies the packaging and deployment. Requires no necessity to maintain servers and operating system administration.
- Easy and efficient auto-scaling.
- High availability and fault tolerance.
- No idle capacity (pay only for what you use).
- Faster time to market and quicker software release.
- Lowers the operational and development costs.
- Utilization of resource and time to develop and improve customer-facing features.
- Reduces the complexity of the software.

10. CONS OF SERVERLESS

- Vendor Lock-in complexity.
- Managing state is relatively complex.
- High latency.
- Testing locally is a risky one.
- Unsuitable for long-term tasks.
- Fear of security breach if it supports multi-tenancy architecture.
- Cost is unpredictable because the count of your code execution is not predefined.

11. POSSIBLE ATTACK VECTORS IN SERVERLESS

- Serverless Computing replaces a number of legacy applications. However, the security concerns and responsibilities get transferred to the cloud provider. At the same time, users are still responsible for running their codes to these cloud platforms. They are listed below:

- **Increased Attack Surface**: As serverless functions consume data from multiple event sources like HTTP API's, message queues, cloud storage, and IoT device communications, the attack vectors will trigger protocols and complex message structures that are very difficult to be examined by a conventional Web Application Firewall (WAF).

- **Malicious Injection Vulnerabilities**: An injection attack may occur from an event that the serverless function calls to execute. A user supplied data should be never trusted blindly. An improper input validation may result in the execution of malicious codes that may crash the application functionality.

- **Denial of Service Attack:** If an attacker finds a way to execute a vast number of serverless events, they may deny or disrupt the service request of a legitimate user. Also, they may influence and consume the cloud computing resources and may misuse it.

- **Security Misconfiguration:** Developers may fail to secure the sensitive data credentials such as tokens, passwords, user ids by calling or invoking them directly in to the function without protecting it and if so, then the probability of misconfiguration is very high.

- **Incorrect Permissions:** Generally, serverless functions are permitted the same permission as a server might get. Least privilege permission is enough for a serverless function to work, as granting a full permission to an application could cause potential risks.

- **Weak Encryption:** An incorrect or weak encryption may reveal the sensitive data of an application. Serverless functions often call the database and the requested resource, via API's and etc. If the connection isn't encrypted, it will open up the function to a potential risk.

- **Component Vulnerabilities:** In a serverless application, every function is built on libraries, and components are provided by a third-party vendor. If there's any known or unknown vulnerabilities in one of the components, the serverless function could be exploited.

- **Lack of proper security testing:** Auditing the security aspects of serverless applications are far more complex than the traditional applications. Even scanners aren't adapted to scan serverless applications yet.

12. TOP 10 CRITICAL RISKS IN SERVERLESS ARCHITECTURE:

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

12.1 (SAS-1) – FUNCTION EVENT-DATA INJECTION

Injection flaws are always topping the OWASP TOP 10 list. Main reason for injection flaw to occur is, when an untrusted user supplied input is passed directly to an interpreter and gets executed or evaluated.

Most serverless architectures provides a multitude of event sources, which can induce the serverless function execution. This set of event sources increases the potential attack surface and introduces complexities when trying to safeguard the serverless function from injection flaws. They are not easy as traditional web environments where developers know which area is vulnerable to injection attacks. (E.g. GET/POST parameters, HTTP Headers etc.)

The tool was used to send a continuous stream of TCP or UDP packets to a single server or website. The recipient, upon having received each packet, will need to process and respond accordingly and eventually, stopping to respond for legitimate requests.

SOME EXAMPLES INCLUDE

- Cloud storage events (e.g. AWS S3, Azure Blob Storage, Google Cloud Storage)
- NoSQL database events (e.g. AWS DynamoDB, Azure cosmos DB)
- SQL database events
- Stream processing events (e.g. AWS Kinesis)
- Code changes and new repository code commits
- HTTP API calls
- IoT device telemetry signals
- Message queue events
- SMS message notifications, PUSH notifications, emails, etc.

TYPES OF INJECTION FLAWS IN SERVERLESS ARCHITECTURE

- Operating System (OS) command injection
- Function runtime code injection (e.g. Node.js/JavaScript, Python, Java, C#, Golang)
- SQL injection
- NoSQL injection
- Pub/Sub Message Data Tampering (e.g. MQTT data injection)
- Object deserialization attacks
- XML External Entity (XXE)
- Server-Side Request Forgery (SSRF)

12.2 (SAS-2) – BROKEN AUTHENTICATION

In a Distributed Denial-of-Service (DDoS) attack, the victim receives a continuous stream of unsolicited messages from multiple sources. This type of attack is similar to the amplified attack. On 1st March 2018, the Hacker News (www.thehackernews.com) reported one of the biggest DDoS attacks to ever take place. This was 1.35 TB humongous, which hit the famous GitHub website (www.github.com).

Serverless applications were built like microservices kind of system design, containing hundreds of distinct serverless functions with its own purpose. Due to this, some may leak public web APIs, while others may serve as a proxy to different functions.

It is necessary to enforce strict authentication mechanisms that will provide proper access control and protection to every relevant function and event type. Example for such an attack would be "Exposing unauthenticated entry point via S3 bucket with public access."

12.3 (SAS-3) – INSECURE SERVERLESS DEPLOYMENT CONFIGURATION

The probability of misconfiguration and critical configuration setting in serverless functions are quite high and it may cause tragic data losses, as the serverless architecture supports and offers many new customer-demanding features, customization and configuration setting for specific needs, tasks, and environment.
It is necessary to safeguard the sensitive data of a function to any unauthorized personnel. Hence, it's better to implement a Strict Access Control List (ACL) and Cloud hardening mechanisms to avoid server misconfiguration issues.

12.4 (SAS-4) – OVER PRIVILEGED FUNCTION PERMISSIONS AND ROLES

It is always wise to follow the principle of "Least Privilege". Technically, this means limiting the permission or privilege for an individual. Granting over privilege to a serverless function paves way to carry out unintended destructive actions. (E.g. Executing System Functions)

12.5 (SAS-5) – INADEQUATE FUNCTION MONITORING AND LOGGING

It is important for an organization to log and monitor the security events in real-time, in-order to detect the intrusion and security breach occurrence effectively. One of the main aspects of serverless architecture is that the Logging and Monitoring resides in the cloud environment, outside of the organizational data center perimeter. Serverless service providers provide efficient logging and monitoring facilities. These logs are in out-of-the-box configuration and aren't suitable for complete security assessment.

For this serverless application, Developers and their DevOps team are required to work together according to their organizational requirements (E.g. collecting the real-time logs from various events, submitting these logs to an effective remote security information and event management system (SIEM)). This will often, at times, require you to store the logs in an intermediary cloud storage service.
The SANS six categories of critical log information paper recommend that the following log reports should be collected:

- Authentication and Authorization reports
- Change reports
- Network activity reports
- Resource access reports
- Malware activity reports
- Critical errors and failures reports

12.6 (SAS-6) – INSECURE 3RD PARTY DEPENDENCIES

A serverless function is a small piece of code that performs a single discrete task. In-order to perform this task, the serverless function relies upon the third-party software packages, open-source libraries, 3rd party API calls, etc. It is recommended to verify the third-party dependencies before importing their code. If there is any known or unknown vulnerability in one of these components, then the serverless function could be exploited.

12.7 (SAS-7) – INSECURE APPLICATION SECRETS STORAGE

As the Serverless applications are expanding at a large scale, the need for storing and maintaining the application secrets are critical. (E.g. Username, Passwords, API keys, Database credentials, Encryption keys, Configuration files, etc.) The most common mistake is storing the application's vital information in plain text within the configuration files or in the database.
If application's sensitive data weren't encrypted, it will be visible for anyone and an attacker can attempt to gain access to the application and cause destruction. (E.g. Any user with "Read" permissions can gain access to these secrets). It is always recommended to encrypt the sensitive data and not to store in plain text.

12.8 (SAS-8) – DENIAL OF SERVICE AND FINANCIAL RESOURCE EXHAUSTION

Denial of Service attack is a common threat vector for both server-based and serverless architecture. DOS attack on serverless application can cause financial and resource unavailability issues.
It is necessary for the developers to strictly define the execution limits when deploying the serverless application in the cloud. (E.g. per execution memory allocation, maximum execution duration per function).

12.9 (SAS-9) – FUNCTIONS EXECUTION FLOW MANIPULATION

Application flow manipulation is common to serverless architecture. The main reason for this vulnerability is that serverless supports multiple types of software. They are microservice design model containing a cluster of discrete functions, coupled and combined together in a specific order, representing the overall application's logic.As functions are chained, invoking a specific function may invoke another function. The order of invocation is critical for achieving the desired logic.

Breaking the application flow will help an attacker to destroy the application's core logic by bypassing access controls, escalating the user privileges, and even it's possible to cause a Denial of Service (DOS) attack.

12.10(SAS-10) – IMPROPER EXCEPTION HANDLING AND VERBOSE ERROR MESSAGES

Serverless applications are not as easy as traditional applications. They involve line-by-line debugging and also are complicated and limited. However, the above factor forces the developers to adopt the use of verbose error messages and in enabling debugging environment variables. They might even forget to remove or hide the code while moving it to the production environment, which is an obvious flaw.

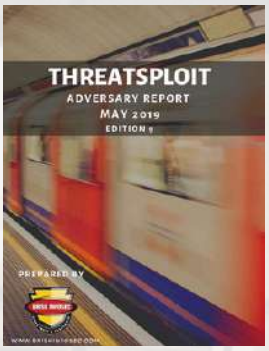
These error messages may reveal some internal codes, logic and technology details about the application being open to everyone. (E.g. verbose error message, stack traces, syntax error.)

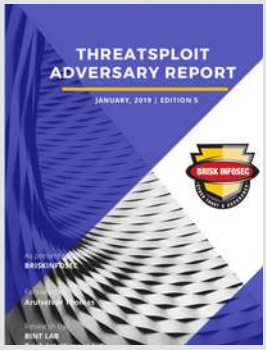
13. CONCLUSION

Serverless computing remains as an exciting new technology. It offers huge business benefits and service to the organizations. There will be many advances in the field over upcoming years. In the following years, it is predicted that serverless computing will be deployed and used by many organizations worldwide. Just like digital took over analog, it is believed that server-based computing will be dominated by serverless computing.

YOU MAY BE INTERESTED ON OUR PREVIOUS WHITEPAPER

YOU MAY BE INTERESTED ON OUR PREVIOUS WORKS





REFERENCES ABOUT BRISKINFOSEC

CASE STUDIES

SOLUTIONS

SERVICES

RESEARCH

COMPLIANCES

BLOGS

This White Paper is proudly presented by

BRISKINFOSEC TECHNOLOGY AND CONSULTING PVT LTD

Feel free to reach us for all your cybersecurity needs
contact@briskinfosec.com | www.briskinfosec.com
|USA|INDIA|UK