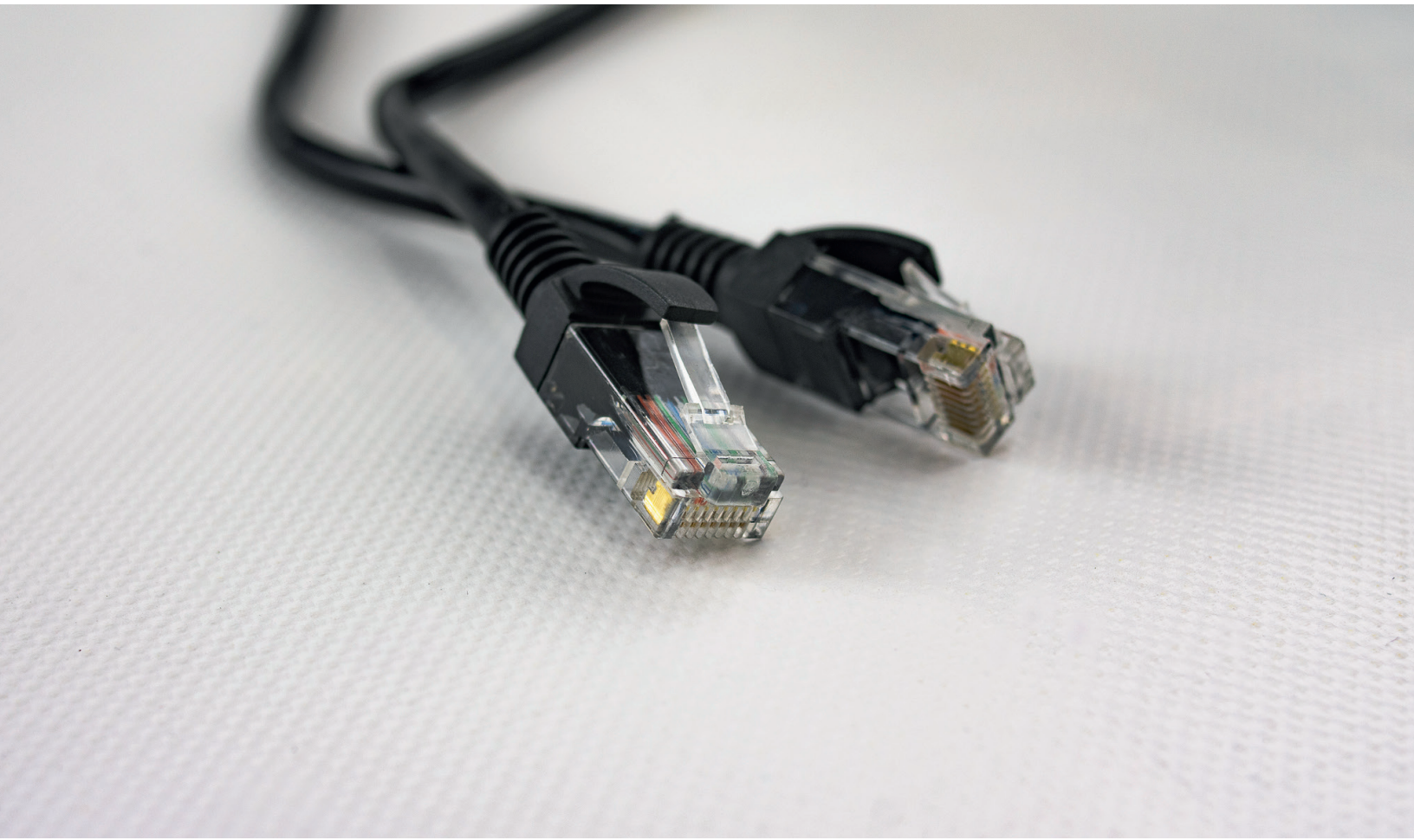
LTI
Let's Solve

A Larsen & Toubro Group Company



Point of View

Understanding and Securing Serverless Computing

by **Varun Kaushik**

Over the past few years, lot of technological innovations such as IaaS, PaaS, and containerization have been developed to resolve the conflict in responsibilities of server and other infrastructure management. Then, FaaS (Function-as-a-Service) – Serverless Computing was innovated to allow developers to focus on code and move away from server and infrastructure's concerns. Of late, it has become the fastest growing cloud service model. The global serverless architecture market size is expected to reach USD 19.84 Billion by 2025, according to a study by Grand View Research, Inc . Big companies like Netflix, Coca-Cola, and Nordstrom have already started using serverless technology in their production environment.

Understanding the Serverless business model

"Serverless" doesn't mean that there is no server or any other infrastructure to run applications, but that these servers and other resources are managed by service providers. Serverless Computing is a cloud computing model that enables developers to focus on their code and innovation, without worrying about infrastructure. The service provider manages all infrastructure management tasks such as server provisioning, patching, maintenance, backup, and so on. There are four important components of any Serverless application -

- **Function** - This is the compute service that executes the code. Developers can write function to carry out any common task. Cloud Service Providers host and maintain these functions, which are invoked through the Internet and other services. Some examples of Functions are AWS Lambda, Azure Functions, Google Cloud Functions.

- **Events** - An event is a JSON (JavaScript Object Notation)-formatted document that contains data for a function to process. Developers determine the structure and contents of the event.

- **Resources** – Functions interact and communicate with other resources such as database and compute, which are hidden behind the compute services functions.

- **Services** – Services are the component of serverless architecture which are useful to solve the certain problems. Some examples of Services are SNS (Simple Notification Service), SQS (Simple Queue Service), SES (Simple Email Service).

Advantages	Disadvantages
Pay for Value – In a serverless computing model, customers pay only for the duration of execution of a function and for the number of functions executed. Customers don't have to pay for an ideal capacity and there is no charge when the customer's code is not running.	**Not for critical application and SLA –** If we see serverless architecture in detail, these applications are completely depending on cloud service providers and other third-party dependencies, which is always a risk for any critical application and SLA.
No Server, No Operational work – Customers don't provision or maintain any server. There is no software or runtime to install, maintain, or administer. Hence, there is no operational work done by customer on servers and other infrastructure services. Customers perform operational tasks on their code like debugging, testing, troubleshooting, etc.	**Decentralization –** Decentralized serverless approach creates its own challenges which creates lot of issues during the troubleshooting of any failure, e.g. latency issue while calling the third-party APIs or other cloud services.
Flexible Scaling – Customers' application can be scaled automatically horizontally or by adjusting its capacity through toggling the units of consumption (e.g. throughput, memory). This is managed by cloud service provider.	**Vendor lock-in –** Most of serverless applications are tightly coupled with cloud platform as those applications use that platform' services and APIs. This problem creates multiple problems including data privacy issue, cost and support, availability of new features, etc.
Fully Managed and Automated High Availability Environment – CSP (Cloud Service Provider) manages complete infrastructure and platform and provides high-availability and fault-tolerance capabilities. Customers don't need to architect for these capabilities as these are provided by default by CSP.	**Security Concerns –** Security is the main concern of serverless architecture, which results in compromise company critical data, leak secrets, denial of services and financial exhaustion, etc.

Here's how to secure it

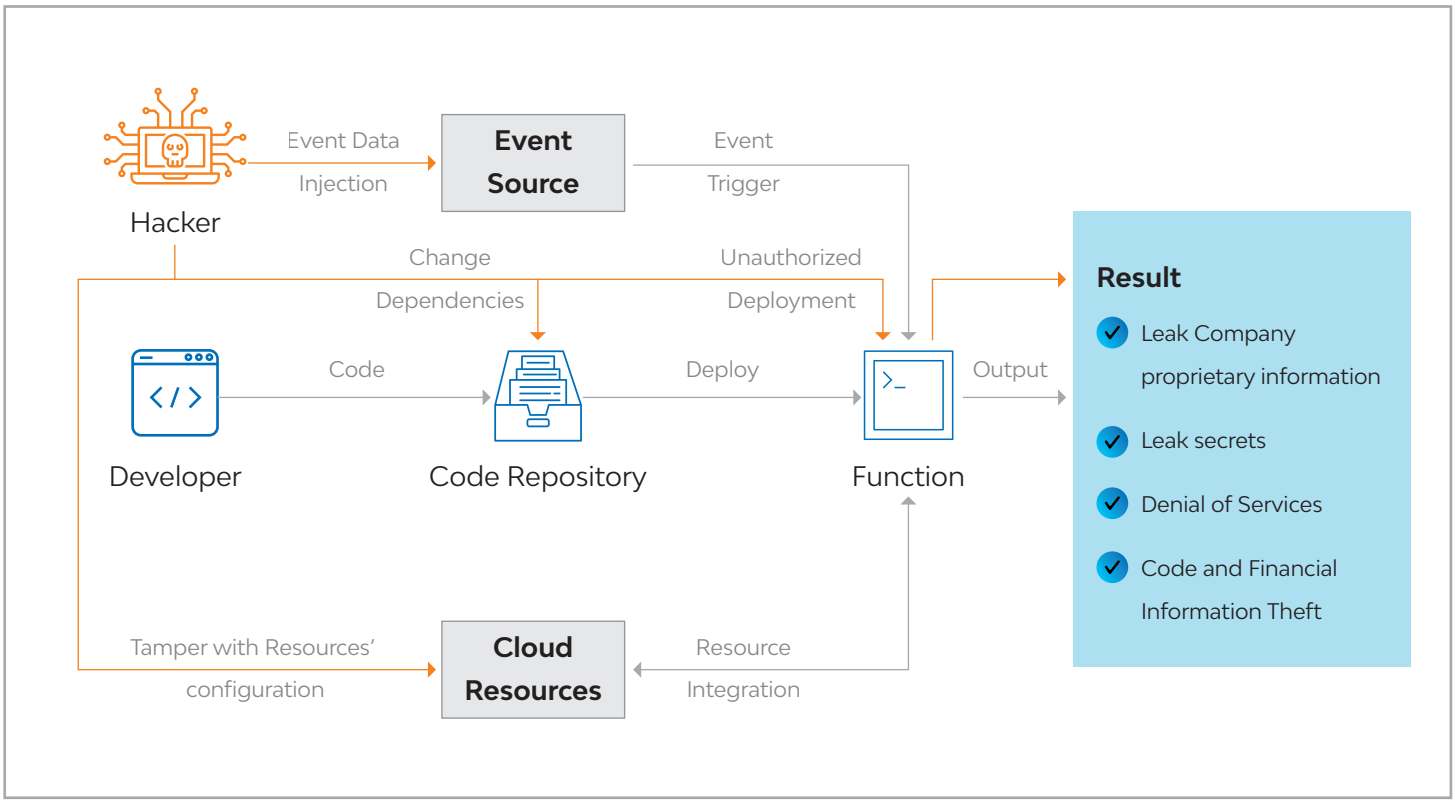
Hackers can target serverless computing components and can inject their code into application to tamper with or steal customer's information:

- **Event Sources** - Hacker can inject untrusted input to an event source, which can trigger the execution of a serverless function and provide attacker-required output or perform any other dangerous or harmful tasks.

- **Cloud Resources** - Serverless applications relying on multiple cloud resources and hacker can easily tamper resources' configuration if weak authentication policy and process is implemented.

- **Function** – Serverless application have multiple functions as a result managing functions' permissions, dependencies, and other security related configuration becomes a tedious task for developer. In this kind of scenarios, developers don't follow the security best practices which provide chance, unintentionally, to hacker.

- **Code Repository** – Hackers can change the code and its dependencies, if code repository is not stored under secured configuration location.



Targets for Hackers in Serverless Application

Security solutions suggested

There is a need of serverless-native protection solution as traditional security methods (e.g. IPS/IDS, Network Firewall,etc.), which are deployed mainly on network and server side, will not be efficient enough to secure the serverless application. Here are some solutions to make serverless application risk-free and more secure -

Control Access (IAM)

Controlling permission and access is a very powerful practice to avoid any kind of attack and expose the function and application code to hackers. Serverless developers must ensure that their code runs with minimum privileges and not give full access on cloud resources.

Don't use "Shift+8"

Serverless developer thinks twice before using "Shift+8 or *". By using this, developer will give complete permission or allowing to access complete resource.

Keep function's code and event source in same repository

Single Repository is very important to stop any injection of untrusted data in an event source. If it is not possible to store function code and event source in the same repository, then the developer must apply robust authentication policies, which provide complete protection to all functions and relevant event source.

Encrypted Secrets/keys

It is very important to store all the secrets (like Keys, Database credentials, encryption keys, etc.) in a secured and in an encrypted format. Customers can use AWS secrets manager and key vault to generate and store all the credentials.

Real-time monitoring and logging

All the cloud providers and other third-party vendors ensure extremely capable monitoring and logging facilities by using their services and tools. Serverless Developer and DevOps team must configure these tools as per their requirements and make sure they capture and identify all the untrusted attempts of malicious activity on functions, cloud resources, and event data. Customers must monitor the permissions and assess as per the corporate and application requirement and keep track of any change. Customers can use cloud providers' native tools like AWS CloudWatch, AWS CloudTrail, Azure monitor, etc.

Check Dependencies Vulnerabilities

Serverless Developers should make sure that their code is not importing code from any obsoleted and vulnerable third-party dependency. For that, they must update records of all the dependencies used in code and keep scanning the dependencies by tools like snyk, pyup, requires.io, dependencies.io, and sourceclear, etc. Developers can use cloud providers' services to identify the potential misconfigurations, for example, AWS Trusted Advisor.

Conclusion

Serverless computing is gaining momentum and multiple organizations are adopting this technology. But, as serverless technology is quite new, developers, architects and DevOps engineer don't have complete knowledge and required expertise to understand the security concerns. To overcome all the challenges and to get all promised benefits, customers must implement new security tools and framework with their serverless computing.

References

- https://www.grandviewresearch.com/industry-analysis/serverless-architecture-market
- https://www.paloaltonetworks.com/resources/whitepapers/the-12-most-critical-risks-for-serverless-applications
- https://aws.amazon.com/compliance/shared-responsibility-model/
- https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html
- https://www.esecurityplanet.com/cloud/serverless-cloud-security.html
- https://dzone.com/articles/serverless-security-risks-and-how-to-mitigate-them
- https://cai.tools.sap/blog/top-10-web-security-vulnerabilities-to-watch-out-for-in-2019/
- https://dashbird.io/blog/serverless-case-study-coca-cola/

About the author

Varun Kaushik

Senior Technical Architect- CIS, LTI

Varun Kaushik has 15+ years' experience in Infrastructure, Cloud, and DevOps Technologies. In his current role, he works as Technical Architect for the Solution Design Center of the Cloud and Infrastructure Practice. He has been involved in multiple datacenter consolidation and application migration projects. He holds Infrastructure and Cloud (AWS and Azure) certificates, and is TOGAF-certified.