Universal Library of Engineering Technology Research Article

ISSN: 3064-996X | Volume 2, Issue 3

Open Access | PP: 100-104

DOI: https://doi.org/10.70315/uloap.ulete.2025.0203018

Ensuring the Security of Serverless Applications Using the Zero Trust Approach

Yevhen Mykhailenko

Software Engineer, PayPal (by Accelon Inc.), Austin, Texas, US.

Abstract

The article examines the problem of securing serverless applications amid the rapid spread of cloud-native approaches and organizations' transition to the code as a function model. The study aims to identify key threats arising from the use of serverless architectures and to justify the application of the Zero Trust concept as a foundational protection model. The topic's relevance is determined by the diminishing efficiency of classic methods of perimeter security because of the ephemeral nature of the function, high speed of deployment, and distributed infrastructure character. The novelty of the paper includes carrying out a systematization of attack vectors in serverless environments and developing a holistic scheme for the application of Zero Trust at all stages of the application life cycle- starting from design and build up to running and monitoring. The analysis has proven that the major vulnerabilities in the serverless model are within identity and access management misconfigurations, an insecure supply chain, and event trigger exploitation. There is an overwhelming presence of both excessive privileges and vulnerable configurations. Make Zero Trust is an objective necessity. In practice, enforce multi-factor authentication and least-privilege design right from the design stage as signed artifacts cryptographically at build time, strict isolation, and egress control at runtime, plus continuous monitoring with automated response at the operations level. This set of measures makes it possible to localize threats and maintain application resilience without sacrificing flexibility and scalability. The article will be helpful to researchers in cloud security, practicing architects, and DevSecOps engineers, as well as executives making decisions about adopting modern protection models.

Keywords: *Serverless, Zero Trust, Cloud Security, Identity and Access, Supply Chain, Vulnerabilities, DevSecOps.*

INTRODUCTION

Serverless architectures, which have become an integral part of cloud transformation, have already moved from the exotic category to the de facto standard: according to the Cloud Native Computing Foundation, in 2024 the share of organizations applying cloud-native approaches reached 89%, and functions and managed events are among the top three most in-demand technologies (Silverthorne & Hendrick, 2025). Gartner analysts predict that by the end of 2025, the use of serverless will become routine practice for more than half of global companies versus 30% a year earlier, meaning growth of about 20 percentage points in a single year, underscoring the rapid spread of the code as a function model (Kasthuri et al., 2024). This popularity is understandable: scale-to-load behavior, per-minute billing, and the absence of OS maintenance duties radically accelerate time-to-market and reduce operational costs.

However, benefits bring a new logic of risk. The ephemeral nature of runtime containers, trigger-driven architecture, and tight integration with managed services minimize classic zonal defense. There are no stable IP segments to surround with a firewall, no long-lived machines where protection agents function correctly, and access control shifts entirely toward IAM roles and temporary tokens. Paradoxically, the same set of advantages that make serverless attractive for DevOps teams complicates defenders' tasks: egress traffic inspection, log analysis, and attack emulation become highly fragmented; the detect—fix loop must fit into minutes, since a function may cease to exist immediately after execution.

Empirical data proves that traditional methods are not working anymore. An Orca Security study revealed more than a billion cloud assets, indicating that 61% of organizations still store root accounts without MFA, and 81% have at least one publicly accessible resource with open ports vulnerable to automated scanning (Orca Security, 2024). These figures perfectly illustrate how much stuck-in-the-culture-of-a-segmented-perimeter the true reality of modern cloud is, where the main element of attack is increasingly not the network but a combination of a mistaken IAM permission, as well as a vulnerable package in the build chain, and an instantly deployed trigger.

Citation: Yevhen Mykhailenko, "Ensuring the Security of Serverless Applications Using the Zero Trust Approach", Universal Library of Engineering Technology, 2025; 2(3): 100-104. DOI: https://doi.org/10.70315/uloap.ulete.2025.0203018.

So the model-agnostic Zero Trust approach is no longer a buzzword, but rather an imperative reality: when each function lives for just some seconds and when request context can change faster than you can update an ACL table, the only stable control point available to implement would be never trust, always verify. That means implementing strictly enforced identity validation for every call, minimization of privileges at the level of roles and service accounts, cryptographic integrity of build artifacts, and continuous event monitoring-these exact elements that can help fill in the gap left by classical predominantly perimeter-oriented protections.

MATERIALS AND METHODOLOGY

The research is based on a comprehensive analysis of academic and industry sources, empirical report data, and practical guides from leading cloud providers. The theoretical foundation is formed from works devoted to the dynamics of serverless architecture adoption and their impact on cloud transformation: according to the Cloud Native Computing Foundation, in 2024, 89% of organizations apply cloud-native approaches, and functions and managed events ranked among the three most in-demand technologies (Silverthorne & Hendrick, 2025). Gartner's forecast confirmed the rapid growth of the code as a function model, showing an increase in the number of companies using serverless from 30% to more than half in one year (Kasthuri et al., 2024). To capture the current threat landscape, Orca Security reports were used, demonstrating systemic gaps in the application of basic protections in cloud environments, including storage of root accounts without MFA and the presence of public resources with open ports (Orca Security, 2024). These data became the empirical basis for framing the problem of traditional security at the periphery.

Methodologically, several Complementary directions were pooled in this study. The first is a systematic review of threat statistics as recorded by industry analysts. Data on the Growth in the number of unauthorized sessions and anomalous account logins (Quist, 2025) together with observations on where prevalences lies in vulnerabilities within event triggers and supply chain (JFrog, 2025; Teemu, 2024) have been used to formulate which key attack vectors are being used. A comparative analysis of serverless architectural features is undertaken in this direction. It includes a study on cold starts and shared runtimes (Chai et al., 2025) that form side channels and new scenarios for data leakage. The third is regulatory and practical protection models that studies: AWS Lambda security guide (AWS, 2025) together with OWASP documentation (OWASP, n.d.) applied here to verify the set of minimum measures that are embedded within a Zero Trust architecture as mandatory elements.

RESULTS AND DISCUSSION

Serverless architectures revolve around a minimal set of managed components: functions run executable code exactly for the time needed to process an event; triggers provide system reactivity, linking invocations to sources such as HTTP gateways, message queues, or object-storage change events; and auxiliary managed services—databases, secret stores, service meshes—assume subsystem resilience and scaling. Together, this forms a Function-as-a-Service model in which the developer operates on business logic, while infrastructure tasks—from container provisioning to autoscaling—remain invisible. The popularity of the approach is corroborated by the annual Cloud Native Computing Foundation survey: in 2024, 89% of organizations reported using cloud-native techniques, and functions, together with containers and Kubernetes, ranked in the top three most commonly used technologies (Silverthorne & Hendrick, 2025). In 2024, there is a shift toward deeper cloud-native adoption: the shares of much and nearly all increased (36% and 24%, respectively), while the intermediate category some decreased (28% vs. 34% in 2023), as shown in Figure 1.

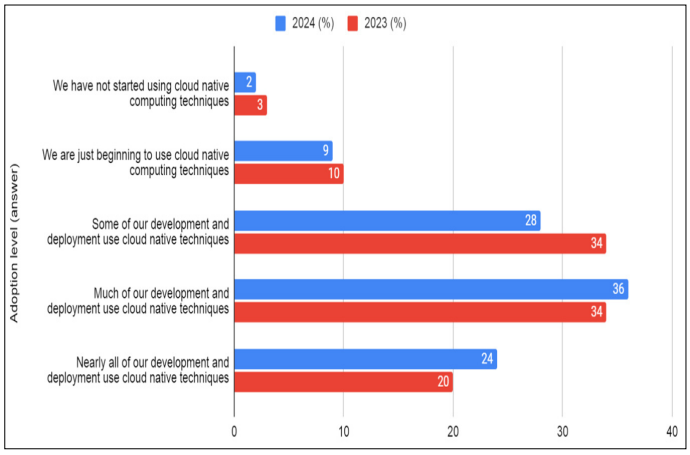

Fig. 1. Cloud-Native Adoption Levels (Silverthorne & Hendrick, 2025)

This design brings obvious benefits. Abandoning permanent virtual machines eliminates OS maintenance costs, and horizontal per-event scaling allows paying only for the actual time the code runs. The market is responding with corresponding growth: The global serverless architecture market size was valued at USD 9.42 billion in 2023 and is projected to grow at a CAGR of 28.2% from 2024 to 2030, as shown in Figure 2 (Grand View Research, n.d.).

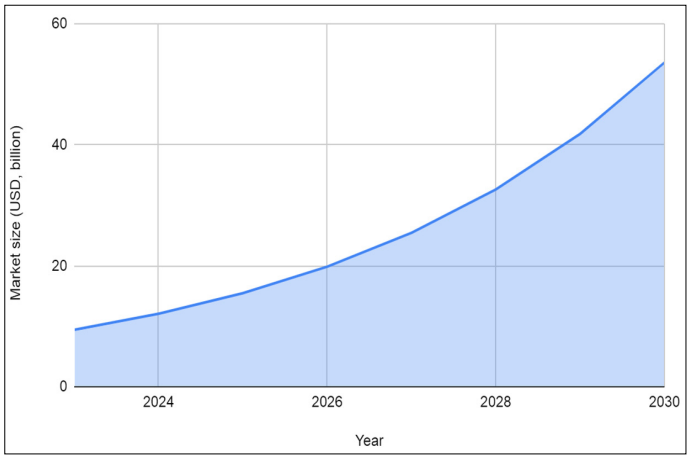

Fig. 2. The global serverless architecture market size (Grand View Research, n.d.)

From a security standpoint, serverless creates a qualitatively new attack surface. Instances live for seconds, so traditional protection agents and periodic scanning lose effectiveness; network boundaries blur, and all decisions are made at the IAM policy level. An erroneously granted role with permissions turns a short-lived function into a springboard for escalation, consistent with OWASP's conclusion that violations of the least-privilege principle remain one of the key causes of application compromise (OWASP, n.d.). The supply chain increases the vulnerability; if an unsecured package gets into the build, it will manifest itself in many short-lived containers until the CI/CD finds out what the problem is. That is why major providers such as AWS emphasize in their guides that strict IAM policy management and cryptographic verification of artifacts are baseline measures to reduce the attack surface in the Lambda environment (AWS, 2025). For AWS Lambda, AWS manages the underlying infrastructure and foundation services, the operating system, and the application platform, as shown in Figure 3.

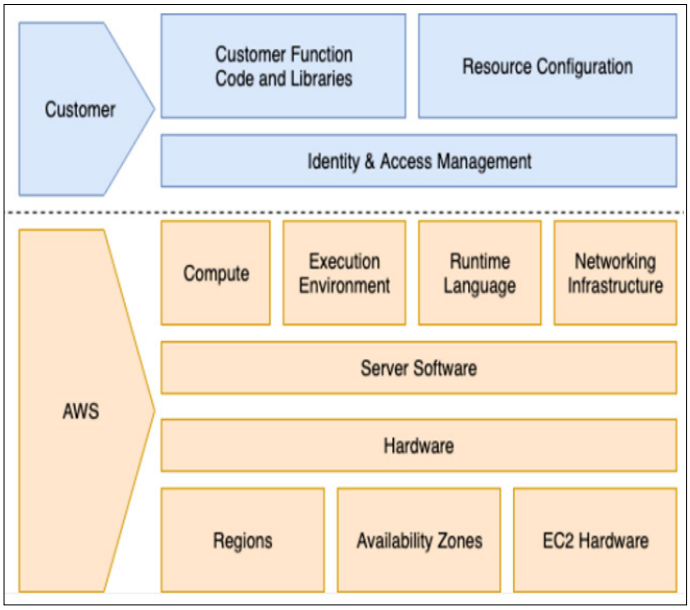

Fig. 3. Shared responsibility model for AWS Lambda (AWS, 2025)

Taken together, these facts show that the code as a function architecture requires a rethinking of classical defense approaches and pushes toward models where trust is minimized at every layer, and verification of identity, code provenance, and invocation context becomes a continuous process.

The shortest route for an attacker in serverless environments remains privilege escalation: all security comes down to how precisely IAM roles are defined. Telemetry from Unit 42 shows that by December 2024 the average organization recorded three times more remote command sessions launched on behalf of function tokens than at the beginning of the year; simultaneously, the number of impossible geographic logins grew by 116%, indicating that excessive privileges of service accounts are becoming a widespread problem (Quist, 2025).

If privileges are the door, event injections are the lock pick. The OWASP DVSA demonstration on AWS Lambda shows how a single unsafe deserialization in a POST-request handler allows JavaScript injection directly into the runtime and exfiltration of environment variables to an external server (Teemu, 2024). Since the trigger layer (API Gateway or S3 event) is attacked, the adversary does not need access to the source code: a specially crafted payload is sufficient, after which each one-off function becomes an entry point.

Secrets are the prime target. In an extensive late-2024 extortion campaign, attackers extracted more than 90,000 credential records from exposed environment-variable files and nearly 1,200 IAM tokens, using them for subsequent blackmail (Quist, 2025). Such statistics elevate the use of specialized vault services from best practice to a mandatory requirement.

The delivery pipeline itself is no less vulnerable. Software Supply Chain State of the Union 2025 indicates that in 71% of organizations, developers are permitted to download packages directly from the internet. On average, a typical company imports 38 new dependencies monthly (JFrog, 2025). Without SLSA signatures and reputation filters, this becomes the easiest stage to tamper with, since once a single library has been poisoned, it gets into the artifact and then all deployed functions thereafter, vulnerability propagates inside enterprises faster than scanners can keep up with.

Finally, the very model of the cold start creates side channels. Research by Ant Group presented at OSDI-2025 showed that even after popular optimizations are introduced, launching a new function takes hundreds of milliseconds, with up to 40% of the time spent on exchanges between runtime layers; the overall process implies caching and container reuse under high contention (Chai et al., 2025). This regimen raises the level of shared memory and devious measurements through which a nearby, less-powered function can assess previously loaded code or data, changing a production optimization into a possible leakage pathway.

All the threats described herein converge on one simple fact: the traditional network perimeter for serverless applications is no longer effective, and, in most cases, it never was. The only reliable barrier is by implementing an end-to-end Zero Trust model where every transaction is validated at levels of identity, artifact integrity, and execution context.

The transition from the threats described to practical measures begins with revisiting the very notion of trust. In serverless environments, the only stable control boundary is a model in which every component—from a function invocation to the movement of a bit of data—is checked for authenticity and policy compliance at the very moment an action is performed. The first pillar of this approach is identity.The ephemeral workflows mandate a multi-factor authentication of every function, they manual or automated, and strictly minimal rights provisioning. Therefore, even if a one-time token gets compromised, the damage would be

localized with no possibility for an escalation chain since the chain gets broken right at the beginning.

The other factor relates to the runtime. The container that executes the function must be booted from a known good layer, pass an integrity check, and be subject to a reduced set of system calls. This level of isolation transforms what was previously a potential beachhead for long-term destructive code within a disposable short-lived instance into a throwaway, managed, and monitored object whose purpose ends with the completion of the request.

Afterward, the net. Because traditional perimeter defense is not applicable, every touch point within the cloud becomes an attack vector. Replace static firewall rules with microsegmentation, end-to-end encryption, and mutual TLS (mTLS) authentication between services, enabling monitoring of every session, regardless of participant location, from inside or outside.

It pertains to utilization. Build artifacts signing and auto-dependency analysis, joined by compile policy enforcement, keep out unverified code artifacts or vulnerable package sources from the supply chain. Another aspect that enhances security comes from input control: validation of the event that initiated invocation, as well as validation of processing logic—injections are excluded, thus cannot take place before business code starts executing.

The fifth pillar is data. In the serverless model, data often moves among managed services, which necessitates encryption at rest and in transit, as well as centralized secret storage. Keys are not distributed via environment variables but are issued strictly on demand, with their lifetime synchronized to the function's execution time. This minimizes the likelihood of leakage and enforces least privilege at the information level.

The final junction is observability. Environments' ephemerality dictates real-time telemetry collection. Only with distributed tracing, streaming log correlation, and ML behavioral pattern analysis can deviations be detected within seconds, automatically contextualized, and response actions initiated even before the attack has brought about tangible damage to the organization. They work by interrelating six facets to minimize trust at every place in the stack and ensure that serverless applications are resilient to contemporary threat types.

Projecting the six pillars of trust into practice, an organization begins work even before the first line of code appears. At the design stage, the core principle is shift left: architectural decisions, threat models, and access-policy requirements are fixed in the specification as rigorously as functional requirements. Schemes of function interactions with managed services are described through threat models, and each proposed permission in a future IAM profile is checked for necessity. Internal security documentation is included in reviews alongside API design to eliminate potential gaps before they reach the repository.

When the automated build begins, infrastructure becomes code. Deployment scripts define every role, environment variable, and network route, and the version control system tracks changes to infrastructure manifests just as it does to source code. An artifact is signed with the build pipeline's key before it gets into the registry; at every deployment invocation, this signature is verified, hence breaking the traditional supply chain and allowing one to detect dependency tampering or unauthorized image changes at publish time. Only approved templates are allowed by validation policies, automatically blocking any configuration outside them; hence, the acceptance pipeline rejects the request while explaining which specific right or variable has caused the refusal.

At execution time, control shifts to the environment itself. Before launching, a function presents a short-lived token obtained through federation with the identity center; its privilege scope is limited to the specific task and the invocation's lifetime. Network communication is encrypted with mutual TLS. Egress traffic goes through a filter that checks the destination of the packet against an allow-listed set of services. If a request is going outside the approved perimeter, then kill the connection before even establishing a session. The runtime isolates the function's system calls by prohibiting operations beyond the necessary minimum. This is, thereby making every instance a throwaway capsule having stiff boundaries within which code does exactly the work it was written for.

The closure happens with continuous monitoring. Authentication logs, function invocations, and network connections are delivered into a central real-time stream; correlation rules relate them to one another, building a chain from the initial request up to the final database write. The anomaly detection system reports unusually high privileged role activations or a spike in outbound traffic to an unfamiliar domain, it triggers automatic function rollback or isolation. This creates a loop through which information flows back into the development pipeline: response time as quality metrics, false positive rates as quality metrics, numbers of rejected deployments as quality metrics, and discovered patterns added to tests and policies of the next sprint. Such a closed loop makes it possible to maintain the never trust, always verify principle at every turn of the serverless application life cycle.

CONCLUSION

The bottom line of the article is that securing serverless applications with the Zero Trust approach requires a fundamental realignment of security principles in tandem with the rapidly growing popularity of serverless architectures. Large-scale adoption of functions as a service amplifies certain vulnerabilities connected to instantaneous deployment and ephemeral runtimes because the classic methodology based on stable perimeters and long periods of monitoring is no longer valid. It, therefore, seems from the

analysis that the main attack vectors comprise mistakes in identity and access management, supply-chain vulnerability, and event trigger exploitation. At the same time, statistics confirm that excessive privileges and unsafe secret storage most often become the sources of compromise.

Under these conditions, Zero Trust is not an additional tool but a mandatory model that embeds security into every stage of the application life cycle. At the design stage, the shift left principle is implemented. Requirements for authentication and least privilege are fixed even before code is written. In the course of implementation, infrastructure as code guarantees supply-chain integrity together with obligatory cryptographic artifact signing. On the execution front, enforced function isolation, mutual attestation, and egress control block any unauthorized interaction. At last, a closed loop of continuous monitoring and automatic response in which security is not an act of assurance at some point but rather a continual process.

Thus, by extending the major vulnerabilities that come with the serverless model to neutralize through ensuring authentication, integrity control, and observability at all levels, Zero Trust can be applied. Never trust, always verify is the ultimate universal mechanism that has the ability to fit into the dynamics of the cloud environment and emerging threats. Therefore, serverless applications preserve their main value – flexibility and scalability when attaining a protection level comparable with or even exceeding traditional architectures.

REFERENCES

1. AWS. (2025). *Security Overview of AWS Lambda*. AWS. https://docs.aws.amazon.com/pdfs/whitepapers/latest/security-overview-aws-lambda/security-overview-aws-lambda.pdf
2. Chai, X., Tan, J., Bie, T., Shen, A., Shen, D., Xing, Q., Song, S., Yang, T., Gao, L., Yu, F., He, Z., Du, D., Xia, Y., Jiao, S., Hu, K., Kang, C., & Chen, Y. (2025). Fork in the Road: Reflections and Optimizations for Cold Start Latency in Production Serverless Systems. *Proceedings of the 19th USENIX Symposium on Operating Systems Design and Implementation*. https://www.usenix.org/system/files/osdi25-chai-xiaohu.pdf
3. Grand View Research. (n.d.). *Serverless Architecture Market Size Report, 2018-2030*. Grand View Research. Retrieved July 20, 2025, from https://www.grandviewresearch.com/industry-analysis/serverless-architecture-market
4. JFrog. (2025). *Software Supply Chain State of the Union 2025*. https://s201.q4cdn.com/814780939/files/doc_presentations/2025/Apr/01/JFrog-Software-Supply-Chain-Report-2025.pdf
5. Kasthuri, M., Yalala, V., Mishra, V., Mehra, A., & Chordiya, S. (2024). *Cloud Trends 2025: Trend report from DMTS Community Of Cloud Technology Unveiling the Future*. https://www.wipro.com/content/dam/nexus/en/lab45/images/cloud-trends-2025-unveiling-the-future-of-cloud-technology.pdf
6. Orca Security. (2024). *State of Cloud Security Report: Uncovering what is lurking in the depths of cloud environments*. https://orca.security/wp-content/uploads/2024/02/2024-State-of-Cloud-Security-Report.pdf
7. OWASP. (n.d.). *A01 Broken Access Control*. OWASP; OWASP. Retrieved July 21, 2025, from https://owasp.org/Top10/A01_2021-Broken_Access_Control/
8. Quist, N. (2025, March 27). *Cloud Threats on the Rise: Alert Trends Show Intensified Attacker Focus on IAM, Exfiltration*. Unit 42. https://unit42.paloaltonetworks.com/2025-cloud-security-alert-trends/
9. Silverthorne, V., & Hendrick, S. (2025). *Valerie Silverthorne, Cloud Native Computing Foundation Cloud Native 2024 Approaching a Decade of Code, Cloud, and Change*. CNCF. https://www.cncf.io/wp-content/uploads/2025/04/cncf_annual_survey24_031225a.pdf
10. Teemu. (2024). *Hacking AWS Lambda Functions - from Theory to Practice*. Nordhero. https://www.nordhero.com/posts/hacking-lambda-functions-from-theory-to-practice/