IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2025

DOI: 10.17148/IJARCCE.2025.141120

Behavioral Anomaly Detection for Real-time Runtime Security in Serverless Computing

Dr. Sachin S. Bere[1], Mrs. Baravkar B.Y[2], Miss.Rutuja S. Shinde[3], Miss.Jyoti J. Chaudhari[4]

Guide, Department of Computer Engineering, Dattakala Group of Institutions Faculty of Engineering, Swami Chincholi, Daund, Pune, Maharashtra, India[1,2]

Student, Department of Computer Engineering, Dattakala Group of Institutions Faculty of Engineering, swami Chincholi, Daund, Pune, Maharashtra, India[3,4]

Abstract: Serverless computing has redefined cloud ap- plication deployment by abstracting infrastructure and enabling on-demand, event-driven execution, thereby en- hancing developer agility and scalability. However, main- taining consistent application performance in serverless environments remains a significant challenge. The dynamic and transient nature of serverless functions makes it difficult to distinguish between benign and anomalous behavior, which in turn undermines the effectiveness of traditional anomaly detection methods. These conventional approaches, designed for stateful and long-running ser- vices, struggle in serverless settings where executions are short-lived, functions are isolated, and observability is limited.

In this first comprehensive vision paper on anomaly detection for serverless systems, we systematically explore the unique challenges posed by this paradigm, including the absence of persistent state, inconsistent monitoring granularity, and the difficulty of correlating behaviors across distributed functions. We further examine a range of threats that manifest as anomalies, from classical Denial- of-Service (DoS) attacks to serverless-specific threats such as Denial-of-Wallet (DoW) and cold start amplification. Building on these observations, we articulate a research agenda for next-generation detection frameworks that ad- dress the need for context-aware, multi-source data fusion, real-time, lightweight, privacy-preserving, and edge-cloud adaptive capabilities.
Through the identification of key research directions and design principles, we aim to lay the foundation for the next generation of anomaly detection in cloud-native, serverless ecosystems.

Keywords: Serverless Computing, Cloud Computing, Edge Computing, Function-as-a-service, Anomaly Detec- tion, DoS, Data Fusion, System Monitoring, Observability.

I. INTRODUCTION

Over the past decade, serverless computing has emerged as a defining paradigm in cloud-native application development [1], [2]. While traditional Infrastructure-as-a-Service (IaaS) and Platform-as-a- Service (PaaS) models offer varying degrees of automa- tion in provisioning and scaling, serverless platforms go further by providing per-request automatic provisioning, transparent scaling, and fine-grained billing based on ac- tual execution time [3], [4]. These advantages eliminate the need for developers to manage runtime environments,instance lifecycles, or idle resource allocation. As a re- sult, serverless computing has seen widespread adoption across a range of domains, including web services [5], machine learning workflows [6]–[8], IoT backends [9], and high-throughput data processing pipelines [10], [11]. This adoption has been facilitated by the development of a diverse ecosystem of serverless platforms, such as commercial offerings AWS Lambda[1], Google Cloud Functions[2], and Azure Functions[3], as well as open- source frameworks like Apache OpenWhisk[4] and Open- FaaS[5], which offer developers increased flexibility in deployment and greater operational control.

While serverless platforms offer simplified deploy- ment and automatic scalability, they also fundamentally reshape the operational model of cloud applications. First, the ephemeral and stateless nature of the execu- tion model introduces unique observability challenges. Empirical studies show that the majority of serverless functions are short-lived, often completing within mil- liseconds to a few seconds [12]–[14]. This temporal brevity presents a fundamental mismatch with traditional monitoring tools, which are designed for long-running, stateful services. Consequently, collecting sufficient run- time telemetry for profiling or debugging becomes a significant challenge.

DOI: 10.17148/IJARCCE.2025.141120

In this paper, we take a step back to examine the unique challenges and open questions surrounding anomaly detection in serverless platforms. Rather than proposing a single detection mechanism, we present a systematic exploration of the fundamental constraints that make anomaly detection in this domain uniquely difficult. We analyze potential sources of anomalies and attack surfaces, including cold start misbehavior, control- plane bottlenecks, unpredictable scheduling latencies, and workload-induced contention. Building on these ob- servations, we articulate a vision for rethinking observ- ability and anomaly detection in serverless systems, one that emphasizes request-level introspection, distributed event correlation, and adaptive telemetry. Our goal is to illuminate a space of design opportunities and research directions toward robust, scalable, and practical anomaly detection solutions across the spectrum of serverless computing, from centralized cloud environments to dis- tributed, resource-constrained edge deployments. *Organization*. The remainder of this paper is organized as follows. In Section II, we present background and motivation, highlighting the limitations of existing mon- itoring and anomaly detection approaches in the context of serverless systems.

In Section III, we characterize the landscape of anomalies and attack surfaces specific to serverless platforms. In Section IV, we outline a forward-looking vision for anomaly detection in server- less computing, identifying key research directions and design considerations. Section V presents emerging op- portunities and outlines several promising directions for future investigation. Section VI examines the practical obstacles to realizing this vision in real-world deploy- ments, including telemetry limitations, data scarcity, gen- eralization barriers, deployment overhead, and evolving threat models. Finally, Section VII concludes the paper by summarizing our key insights, reiterating our vision for next-generation anomaly detection, and outlining its broader implications for robust serverless ecosystems.

II. THEORETICAL BACKGROUND

Serverless computing represents a fundamental shift in how cloud applications are built and operated. By abstracting away server management, resource provisioning, and scaling, serverless platforms such as AWS Lambda, Google Cloud Functions, and Azure Functions allow developers to focus exclusively on writing event- driven functions. To comprehend the operational mechanisms of a serverless platform, we first examine its architecture and the typical control flow during function invocation. Figure 1a illustrates a generalized four-layer architecture. Figure 1b then illustrates how this architecture is concretely realized within Apache Open Whisk.

In general, the 4-layer architecture of a serverless platform comprises the following components:

Coordination layer: Handles external interactions and exposes endpoints (e.g., REST APIs, SDKs) through which users deploy and invoke functions.
Orchestration layer: Responsible for request rout- ing, scheduling, and load balancing. This layer includes components such as the controller in Open- Whisk, which manages invocation dispatching.
Encapsulation layer: Manages compute resources, typically a pool of containers or lightweight VMs, that execute functions in isolated environments.
Infrastructure layer: Supports low-level resource provisioning, network virtualization, and storage services, typically abstracted by container runtimes and orchestration platforms such as Kubernetes.

When a function invocation is received, it first passes through the coordination layer, which includes the API gateway (e.g., Nginx[6] in OpenWhisk). The request is then forwarded to the *controller* in the orchestration layer, which temporarily buffers the invocation and at- tempts to schedule it onto an available runtime in the encapsulation layer (e.g., an invoker in OpenWhisk), responsible for provisioning or reusing a containerized environment to execute the function.
In serverless computing, a *warm container* refers to a pre-initialized container ready to handle incoming re- quests immediately. In contrast, a *cold start* occurs when no warm container is available, requiring the system to provision a new instance and initialize the runtime and dependencies.

To ensure efficient operation, serverless platforms also enforce runtime control mechanisms. Each function invocation is subject to a *timeout limit*, which bounds its maximum execution duration to prevent indefinite re- source consumption (e.g., Amazon lambda max timeout limit 15 mins). Additionally, most platforms apply *rate limits*, capping the number of concurrent invocations per user account or region to ensure fair resource allocation and system stability.
Analyses of real-world function traces [12], [14] demonstrate that many serverless functions execute within just a few milliseconds or seconds (notably, 85% of all tasks complete in under one minute [22]), and their invocation patterns can be highly sporadic, i.e., exhibiting bursts of activity within a single minute followed by extended idle periods lasting several minutes. This temporal volatility reflects the inherently short-lived and invocation-driven nature of serverless workloads, a behavior we refer to as *ephemeral* [14]. Such behavior introduces unique challenges for traditional anomaly detection methods, as discussed below.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2025

DOI: 10.17148/IJARCCE.2025.141120

III. LITERATURE SURVEY

In conventional cloud environments, applications are deployed on long-running virtual machines (VMs) or containers, which persist over time and expose a rich set of resource and performance metrics, such as CPU utilization, memory usage, disk I/O, and network through- put. These metrics are typically collected by monitoring agents, sidecar containers, or telemetry tools such as Prometheus[7], cAdvisor[8], or Datadog[9]. For instance, Prometheus periodically scrapes the */metrics* endpoint (e.g., every 15 seconds) and stores the results in a time-series database for subsequent analysis.

Anomaly detection in such environments often re- lies on statistical thresholding, time-series modeling, or supervised machine learning techniques trained on historical telemetry data [23]–[27]. These approaches assume stable and continuous execution environments, consistent data collection intervals, and persistent access to system-level signals. While observability tools such as Prometheus can be configured to scrape metrics from short-lived services, doing so reliably at serverless timescales remains challenging. Serverless platforms abstract away the infrastructure layer, offering no direct access to node-level metrics or runtime context. Even when metrics or logs are made available (e.g., via AWS CloudWatch), they are typically delayed, coarse-grained, and insufficient for fine-grained anomaly detection in systems. Consequently, traditional anomaly detection methods struggle in serverless environments, where execution is transient, metric availability is sparse, and system-level visibility is limited.

In addition, serverless workloads are highly bursty and non-stationary [14], with functions triggered by external events that can cause abrupt spikes in invocation rates. Cold starts introduce further variability in response times and are often difficult to distinguish from genuine performance anomalies. Since scheduling, placement, and resource reuse are entirely managed by the platform, developers lack both visibility into and control over the underlying orchestration mechanisms. These constraints hinder the applicability of traditional detection signals, such as CPU saturation or memory leaks, which may be absent or unobservable in a serverless context.

Motivation for rethinking anomaly detection. The unique characteristics of serverless computing motivate a fundamental rethinking of how anomalies should be
detected and diagnosed. From the perspective of server- less application developers or providers (i.e., users who deploy functions on a serverless platform), detection must shift from infrastructure-centric monitoring to a more *request-centric* inference approach. Since direct observations of containers or hosts are typically hidden and only accessible to platform operators, detection systems must instead rely on behavioral signatures – such as function invocation patterns (e.g., function A triggered after function B), response time fluctuations, cold start frequency, and inter-function delays. This calls for lightweight, data-efficient, and *context-aware* techniques capable of extracting meaningful insights from fragmented telemetry.

From the perspective of the platform operator, or in custom platforms where deeper instrumentation is possible (e.g., Open Whisk, Open FaaS), access to low- level resource metrics (e.g., CPU, memory, I/O) along- side unstructured logs from internal components (e.g., controller traces, invoker logs, platform events) enables a more fine-grained view of system behavior. This opens the opportunity to develop log-data fusion mechanisms that correlate function-level behaviors with underlying infrastructure dynamics, thereby improving the accuracy and timeliness of anomaly detection. In addition, the distributed and event-driven nature of serverless workloads, especially in cloud–edge deployments, introduces significant complexity for anomaly detection. Functions may execute across heterogeneous nodes with varying performance characteristics, network conditions, and geographic locations. Moreover, anomalies can arise not just from isolated failures but from cascading issues across microservices, message queues, or third-party APIs (e.g., Mailgun API[10]). In such environments, centralized detection approaches are often infeasible due to high latency, limited visibility, and scalability concerns. These realities call for detection mechanisms that are decentralized, capable of operating close to the execution point, while remaining low-latency and adaptive to dynamic execution contexts.

In essence, the operational realities of serverless computing present a profound architectural mismatch for conventional anomaly detection. This disconnect under- scores the urgent need for a new generation of detection methodologies, ones inherently designed for ephemeral, opaque, and highly dynamic environments. Such methods must bridge the visibility gaps for developers and provide robust, fine-grained insights for operators. In the following sections, we characterize the unique challenges and emerging anomaly patterns in this space, and present a vision for developing more suitable, scalable, and adaptive detection techniques aligned with the serverless paradigm.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2025

DOI: 10.17148/IJARCCE.2025.141120

IV. THREAT LANDSCAPE IN SERVERLESS COMPUTING

Anomalies, defined as observations deviating significantly from expected behavior and potentially indicating underlying issues [28], can manifest in various forms within serverless systems. These include *point anomalies* (e.g., a single invocation exhibiting abnormal latency), *collective anomalies* (e.g., a sequence of degraded executions), or *contextual anomalies* (e.g., failures that occur only under specific workload conditions, such as peak traffic) [29]. In serverless environments, such anomalies commonly appear as increased cold start latency, reduced throughput, elevated error rates, or resource exhaustion, leading to degraded quality of service (QoS), violations of service-level agreements (SLAs), denial of service, and, in some cases, system-wide interruptions or outages. The architectural principles of serverless computing significantly reshape traditional system vulnerabilities and threat models [30], [31]. We categorize the key threats and their manifestations into two primary areas:
(i) *operational vulnerabilities*, which arise from the inherent dynamics of serverless resource management and affect availability and performance, and (ii) *adversarial threats*, which originate from malicious actors exploiting these dynamics for financial, security, or service disruption goals.

Operational vulnerabilities: internal stressors to availability and performance
These threats primarily arise from the inherent characteristics of serverless platforms, such as unpredictable execution, resource management complexities, and dependencies on shared backend services. Anomalies in this domain typically reflect system inefficiencies or degradation.
Cold start latency amplification: The overhead of initializing new function instances (cold starts) can severely impact perceived performance and violate Quality of Service (QoS) guarantees. Un- der bursty workloads, a surge of concurrent cold starts may amplify latency across the system, especially for latency-sensitive functions or edge deployments [32], [33].
Resource contention and noisy neighbor effects: Despite containerization and runtime isolation, serverless functions often share underlying compute infrastructure (e.g., CPU, memory, I/O).
Orchestration delay and queuing collapse: Serverless platforms rely on orchestration layers (e.g., controllers, load balancers, and schedulers) to manage invocation routing and instance life- cycle [18]. Centralized bottlenecks or inefficient scheduling decisions, particularly under diurnal or regionally correlated bursts, can introduce significant dispatch latency. In extreme cases, this may trigger queuing collapse, cascading timeouts, or systemic unresponsiveness.
Backend and dependency failures: Serverless functions frequently depend on external services, such as storage, message brokers, databases, or third-party APIs. Failures or degraded performance in these dependencies propagate to the function layer, manifesting as elevated error rates or long-tail latencies. Due to the decoupled nature of serverless architectures, identifying root causes across service boundaries remains a persistent challenge.
Adversarial threats: external exploitation of server- less weaknesses.
Beyond system-internal sources of performance degradation, serverless platforms are increasingly vulnerable to adversarial threats. Malicious actors exploit the architectural characteristics and operational dynamics of serverless systems to induce service disruption, escalate costs, or evade detection. These threats often mirror or amplify benign anomalies, blurring the boundary between fault and attack. As a result, anomaly detection mechanisms must account for both accidental failures and deliberate manipulation of the control and data planes.

Denial-of-Service (DoS) and Denial-of- wallet (DoW) attacks: While serverless platforms auto-scale to absorb traffic, a sustained high volume of requests targeting publicly exposed functions (e.g., via API Gateways) can still saturate configured concurrency limits [36]. This can lead to legitimate requests being queued or dropped, effectively denying service to legitimate users, especially if cold start overheads prevent rapid scaling. More subtly, DoW attacks [31], [37] aim to silently drain tenant budgets by repeatedly invoking costly functions, e.g., inference for machine learning (ML) models or data-intensive workflows. Even low-rate, persistent invocations can accumulate substantial cost, succeeding without triggering rate limits or violating correctness constraints.

Trigger abuse and event injection: Serverless functions, by their event-driven nature, rely on di- verse activation triggers like HTTP requests, object storage events, and message queues. Adversaries can exploit misconfigured, publicly exposed, or weakly authenticated triggers to invoke functions out of context or with crafted payloads [30]. For example, a malicious file uploaded to a monitored storage bucket could trigger sensitive data processing, or spoofed webhooks/timers might initiate unintended execution. When functions are chained in workflows (e.g., AWS Step Functions), such trigger abuse can cascade, amplifying the attack's impact across dependent services.
effectively denying service to legitimate users, especially if cold start overheads prevent rapid scaling. More subtly, DoW attacks [31], [37] aim to silently drain tenant budgets by repeatedly invoking costly functions, e.g., inference for machine learning (ML) models or data-intensive workflows. Even low-rate, persistent invocations can accumulate substantial cost, succeeding without triggering rate limits or violating correctness constraints.

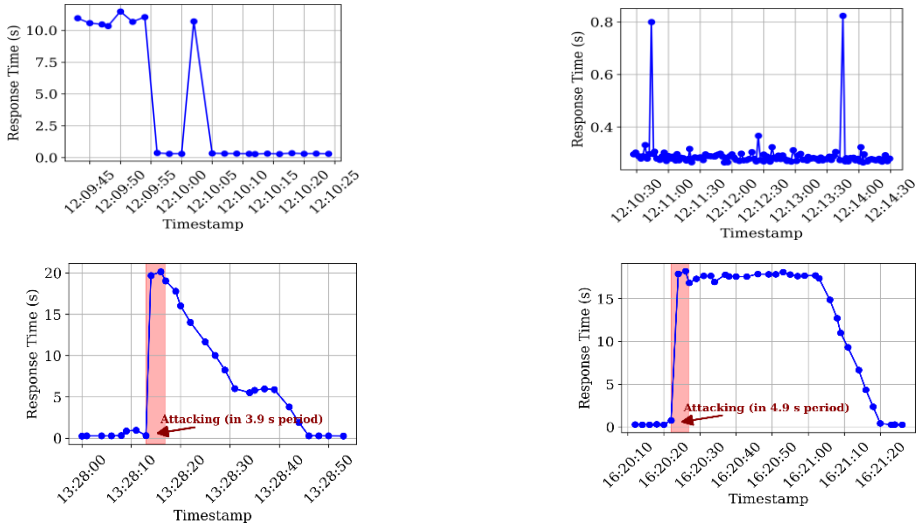
IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2025

DOI: 10.17148/IJARCCE.2025.141120

Evasion and obfuscation tactics: The ephemeral and stateless nature of serverless functions can be exploited by attackers to evade detection and forensic analysis. Malicious logic can be segmented across multiple invocations or injected via environ- ment variables, making it harder to identify a com- plete attack signature in any single event [38], [39]. Due to asynchronous logging pipelines or sampling in highly concurrent environments, rapid-invocation attacks or quick bursts of malicious activity might escape full visibility in the platform's telemetry.



V. CONCLUSION

Serverless computing is rapidly transforming cloud application development, offering immense scalability and simplified deployment through automatic scaling and fine-grained billing. In these dynamic, event-driven environments, even brief disruptions, such as latency spikes, execution failures, or unexpected cost anomalies, can severely impact user experience and operational effi- ciency. As a result, anomaly detection becomes essential for continuously monitoring system behavior, identifying deviations from expected norms, and enabling timely mitigation of emerging issues.

However, the intrinsic characteristics of server- less platforms pose significant challenges to effective anomaly detection. In this vision paper, we meticulously identified these specific challenges, including the ab- sence of persistent contexts, abstracted runtimes, event correlation difficulties, and monitoring granularity gaps. We also explored the full spectrum of operational vul- nerabilities and novel adversarial threats, from Denial of Service and Denial-of-Wallet attacks to more sophisti- cated exploits such as cold start amplification.

Building on these insights, we articulated a com- pelling vision for next-generation anomaly detection frameworks. Our proposed research agenda centers on techniques that are context-aware, leverage multi-source data fusion, operate in real-time, prioritize privacy, and adapt to edge–cloud deployments.

REFERENCES

[1]. Y. Nam, P. Trirat, T. Kim, Y. Lee, and J.-G. Lee, "Context- aware deep time-series decomposition for anomaly detection in businesses," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 330–345, Springer, 2023.

[2]. J. Forough, M. Bhuyan, and E. Elmroth, "Detection of VSI-DDoS Attacks on the Edge: A Sequential Modeling Approach," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, ARES '21, (New York, NY, USA), Association for Computing Machinery, 2021.

[3]. N. Jha, S. Lin, S. Jayaraman, K. Frohling, C. Constantinides, and D. Patel, "Llm assisted anomaly detection service for site reliability engineers: Enhancing cloud infrastructure resilience," 2025.

[4]. Y. Lee, J. Kim, and P. Kang, "Lanobert: System log anomaly detection based on bert masked language model," *Applied Soft Computing*, vol. 146, p. 110689, 2023.

[5]. S. Chen and H. Liao, "Bert-log: Anomaly detection for system logs based on pre-trained language model," *Applied Artificial Intelligence*, vol. 36, no. 1, p. 2145642, 2022.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 11, November 2025
DOI: 10.17148/IJARCCE.2025.141120

[6]. S. Fang, X. Pan, S. Xiang, and C. Pan, "Meta-msnet:Meta- learning based multi-source data fusion for traffic flow predic- tion," *IEEE Signal Processing Letters*, vol. 28, pp. 6–10, 2020.
[7]. E. Bareinboim and J. Pearl, "Causal inference and the data-fusion problem," *Proceedings of the National Academy of Sciences*, vol. 113, no. 27, pp. 7345–7352, 2016.
[8]. A. Makarenko, A. Brooks, T. Kaupp, H. Durrant-Whyte, and Dellaert, "Decentralised data fusion: A graphical model ap- proach," in *2009 12th International Conference on Information Fusion*, pp. 545–554, IEEE, 2009.
[9]. D. Koller and N. Friedman, *Probabilistic graphical models:principles and techniques*. MIT press, 2009.
[10]. R. Dwivedi, D. Dave, H. Naik, S. Singhal, R. Omer, P. Patel.