Borneo International Journal eISSN 2636-9826; Vol. 9 (2); 2026; 23-34
Published by *Majmuah Enterprise*
www.majmuah.com

Serverless Security Misconfigurations in Event-Driven Architectures: A Systematic Review of Emerging Threats and Modern Defense Mechanisms

MARHAKIM MOHAMAD MOKHTAR and MOHAMAD FADLI BIN ZOLKIPLI

School of Computing, College of Art and Science, Universiti Utara Malaysia (UUM), 06010 Sintok, Kedah, MALAYSIA
Email: marhakimm@gmail.com m.fadli.zolkipli@uum.edu.my| Tel: +60194903221 |+60177247779 |

Received: May 09, 2026
Accepted: May 14, 2026
Online Published: June 04, 2026

Abstract

The proliferation of serverless computing and Function-as-a-Service (FaaS) architectures has fundamentally transformed cloud-native application development, enabling unprecedented scalability and operational efficiency. However, the abstraction of underlying infrastructure has introduced a distinct attack surface characterized by critical security misconfigurations, inadequate runtime isolation, and complex privilege escalation vectors. This systematic review examines emerging security threats and modern defense mechanisms within event-driven serverless architectures, specifically focusing on AWS Lambda and Azure Functions environments. Through comprehensive analysis of recent literature (2020–2025) and industry reports, this study identifies five critical protection domains: runtime isolation vulnerabilities, Denial-of-Wallet (DoW) attacks, supply chain risks in function dependencies, over-privileged Identity and Access Management (IAM) configurations, and cross-tenant data leakage. The analysis reveals significant gaps in Backend-as-a-Service (BaaS) orchestration layer security and highlights the transition toward lightweight Trusted Execution Environments (TEEs), microVM-based isolation (Firecracker), and agentless monitoring solutions. This review evaluates the efficacy of emerging defenses including WebAssembly sandboxing, artificial intelligence-driven anomaly detection, and zero-trust architectures in mitigating sophisticated attacks while maintaining serverless performance characteristics. This review contributes a holistic security framework that addresses the intersection of event-driven workflows and serverless misconfigurations, providing actionable insights for practitioners and researchers navigating the evolving threat landscape of 2024–2025. The findings underscore the necessity for defense mechanisms that balance security rigor with the energy efficiency and cold-start latency requirements inherent to serverless paradigms.
Keywords: serverless security, FaaS vulnerabilities, event-driven architecture, security misconfigurations, Trusted Execution Environments, Denial-of-Wallet attacks, cloud-native security

1. Introduction

1.1 Research Background

The architectural paradigm of serverless computing has experienced exponential adoption, with organizations migrating from monolithic infrastructures to event-driven, Function-as-a-Service (FaaS) models (Buyya et al., 2018). Platforms such as AWS Lambda, Azure Functions, and Google Cloud Functions abstract server management, allowing developers to focus solely on code execution triggered by discrete events. This shift promises enhanced scalability, reduced operational overhead, and granular cost structures based on actual execution time (Eivy, 2017). However, the dissolution of perimeter-based security models in serverless environments necessitates a fundamental reevaluation of defensive strategies (Nastic et al., 2017). Event-driven architectures (EDAs) compound these security challenges by introducing complex asynchronous workflows where functions interact through message queues, event buses, and API gateways (Rahman et al., 2022). While this decoupling enhances system resilience, it simultaneously obscures data flow visibility and expands the attack surface through inter-service communication channels. The dynamic nature of serverless provisioning—characterized by ephemeral containers, cold starts, and rapid scaling—creates unique vulnerabilities distinct from traditional virtual machine or container-based deployments (Wang et al., 2018).

1.2 Problem Statement

Recent security analyses indicate that misconfigurations represent the predominant vulnerability vector in serverless deployments, with the Open Web Application Security Project (OWASP, 2021) identifying "Security Misconfiguration" and "Vulnerable and Outdated Components" as critical risks in cloud-native applications. The serverless model inherently complicates security governance: functions often operate with excessive IAM privileges, third-party dependencies introduce supply chain vulnerabilities, and the shared-responsibility model creates ambiguity regarding security obligations between cloud providers and consumers (Datta et al., 2022). Emerging threat vectors have evolved beyond traditional injection attacks to exploit the economic models of serverless computing. Denial-of-Wallet (DoW) attacks weaponize the pay-per-invocation billing structure, forcing resource exhaustion to generate prohibitive costs (Wang et al., 2023). Additionally, runtime isolation failures in multi-tenant environments enable

cross-function data leakage and privilege escalation, while event injection attacks target the asynchronous messaging layers that coordinate serverless workflows (Rahman et al., 2022).

1.3 Research Objectives and Scopes

This systematic review addresses the following research questions:

1. What are the predominant security misconfigurations in current event-driven serverless architectures?
2. How have threat vectors evolved with the adoption of FaaS platforms (2024–2025)?
3. What modern defense mechanisms effectively mitigate these risks without compromising serverless performance characteristics?

The scope encompasses AWS Lambda and Azure Functions as representative platforms, analyzing peer-reviewed literature, industry whitepapers, and documented security incidents. This review specifically examines solutions proposed within the last five years, with emphasis on innovations in hardware-based isolation, agentless monitoring, and zero-trust implementations (Kumari et al., 2025).

1.4 Significance of the Study

As organizations increasingly deploy sensitive workloads in serverless environments—ranging from financial transaction processing to healthcare data analytics—the security implications of misconfigurations escalate proportionally (Kumari et al., 2025). This review synthesizes fragmented research across runtime security, network protection, and orchestration layers to provide a unified security framework. By distinguishing between platform-provider responsibilities and tenant-side configurations, this work enables practitioners to implement defense-in-depth strategies tailored to the serverless paradigm.

2. LITERATURE REVIEW

2.1 Runtime Isolation and Multi-Tenancy Security

The foundational security challenge in serverless computing resides in runtime isolation mechanisms. Early FaaS implementations relied on container-based isolation, which demonstrated susceptibility to side-channel attacks and privilege escalation (Wang et al., 2018). Recent literature (2023–2025) documents a paradigm shift toward microVM architectures, exemplified by AWS Firecracker, which provides lightweight virtualization with reduced attack surfaces compared to traditional containers (Agache et al., 2020; Kumari et al., 2025). *Hardware-Assisted Isolation.* Contemporary research emphasizes Trusted Execution Environments (TEEs) such as Intel Software Guard Extensions (SGX) and AMD Secure Encrypted Virtualization (SEV) as critical enablers for confidential computing in serverless contexts. These technologies establish secure enclaves that protect function execution from both infrastructure administrators and co-tenants (Kumari et al., 2025). However, significant challenges persist regarding cold-start latency overhead and memory constraints within enclaves, necessitating optimization frameworks that balance security guarantees with performance requirements (Ahmad et al., 2021). *WebAssembly Sandboxing.* Emerging as a lightweight alternative to hardware-based isolation, WebAssembly offers near-native performance with language-agnostic sandboxing. Recent proposals integrate WebAssembly with TEEs to create hybrid isolation models, leveraging WebAssembly's fine-grained capability-based security alongside hardware encryption (Kumari et al., 2025). This approach addresses the cold-start dilemma by maintaining smaller memory footprints compared to full VM isolation while providing cryptographic assurance of code integrity (Goltzsche et al., 2019).

2.2 Comparative Security Architectures: AWS Lambda versus Azure Functions

While serverless platforms share common architectural principles, significant security differentiation exists between market leaders AWS Lambda and Azure Functions. Table 1 presents a systematic comparison of security features, isolation mechanisms, and configuration management approaches across both platforms.

Comparative Security Analysis of AWS Lambda and Azure Functions

Security Domain	AWS Lambda	Azure Functions	Critical Implications
Runtime Isolation	Firecracker microVMs (Linux KVM-based); dedicated microVM per function execution with minimal device model (Agache et al., 2020)	Container-based isolation with options for App Service Plan (dedicated VMs) or Consumption Plan (shared infrastructure); Windows/Linux container pools	Lambda's microVMs provide stronger kernel-level isolation than Azure's container sharing, though both remain vulnerable to side-channel attacks without TEE supplementation (Kumari et al., 2025)
Cold Start Security	SnapStart (Java) creates encrypted Firecracker snapshots; ephemeral	Pre-warmed workers in Premium Plan reduce cold starts but increase data	SnapStart reduces initialization latency but introduces snapshot integrity verification

	execution context with 15-minute max retention	remanence risk; container reuse patterns vary by SKU	requirements; Azure's longer-lived containers increase cross-tenant data leakage potential (Wang et al., 2018)
Identity & Access Management	AWS IAM roles with fine-grained permissions; temporary credentials via STS (15 min-12 hr); Resource-based policies for cross-account access	Azure AD integration; Managed Identities (System/User-assigned); Role-Based Access Control (RBAC) with built-in roles; OAuth 2.0/OpenID Connect support	Azure's enterprise identity integration simplifies SSO but increases attack surface via directory synchronization; Lambda's IAM complexity often leads to over-privileged functions (ISACA, 2025)
Networking & VPC	VPC integration via Elastic Network Interface (ENI) cold start latency (~10s); PrivateLink for private API invocation; API Gateway WAF integration	Virtual Network (VNet) integration with regional endpoints; Private Endpoints for Azure Storage/Queue; Application Gateway WAF; Network Security Groups (NSGs)	Lambda's ENI initialization adds security provisioning latency; Azure's VNet integration offers more granular subnet isolation but requires complex peering configurations (Rahman et al., 2022)
Encryption Models	At-rest: KMS (AES-256); In-transit: TLS 1.2+; Client-side encryption optional; Environment variables encrypted by default	At-rest: Azure Storage Service Encryption (SSE); In-transit: TLS 1.2+; Azure Key Vault integration; Managed keys vs. Customer-managed keys (CMK) options	Both platforms lack mandatory client-side encryption for event payloads; Lambda's KMS integration more mature for function-specific secrets, Azure Key Vault superior for enterprise key rotation (Kumari et al., 2025)
Monitoring & Observability	CloudWatch Logs/Metrics/X-Ray; AWS CloudTrail for API auditing; AWS Security Hub for compliance; Lambda Extensions for agentless monitoring	Application Insights; Azure Monitor; Log Analytics workspaces; Defender for Cloud integration; Event Grid for security event routing	Azure's native SIEM integration (Sentinel) provides superior correlation across enterprise workloads; Lambda's X-Ray offers better distributed tracing granularity for microservices (Soni & Sehgal, 2024)
Event Source Security	S3, SQS, SNS, EventBridge, API Gateway, DynamoDB Streams; Resource policies validate event source ARN; Event filtering at source	Event Grid, Storage Queues, Service Bus, Cosmos DB triggers; WebHook validation; Event Grid input validation policies	EventBridge schema validation reduces injection risks versus Azure Event Grid's broader publisher trust model; both vulnerable to event spoofing without strict source validation (Datta et al., 2022)
Deployment & Supply Chain	AWS CodeDeploy, CloudFormation, SAM; Signer for code signing; ECR for container images; Inspector for vulnerability scanning	Azure DevOps, ARM templates, Bicep; Azure SignTool; Container Registry with ACR Tasks; Defender for Container Registries	Lambda's IAM-based deployment credentials often hardcoded in CI/CD; Azure's service principal management reduces credential leakage but increases lateral movement risk if compromised (Lynn et al., 2022)
Concurrency & DoW Protection	Reserved concurrency limits; Provisioned Concurrency (costly); Billing alerts only; No native cost-based throttling	Function-level concurrency limits; App Service Plan instance caps; Budget alerts via Azure Cost Management; Rate limiting at API Management layer	Neither platform implements automatic DoW detection; Azure's API Management offers more granular rate limiting, while Lambda's concurrency controls are easier to misconfigure (Wang et al., 2023)
Private Package Integration	VPC Endpoints for CodeArtifact; Private npm/PyPI via CodeArtifact; Layers for shared dependencies	Private NuGet/npm feeds via Azure Artifacts; Private container registries; Mountable File Shares for persistent dependencies	Both platforms expose supply chain risks through public package managers; Azure's tighter Visual Studio integration encourages dependency auto-

			updates that may introduce vulnerabilities (Kumari et al., 2025)

2.3 Event-Driven Architecture Vulnerabilities

The asynchronous, decoupled nature of event-driven serverless architectures introduces distinct vulnerability classes separate from synchronous request-response models. Research identifies event injection attacks—where malicious payloads traverse message queues (Amazon SQS, EventBridge) or pub/sub systems (Apache Kafka, Amazon SNS)—as critical vectors for lateral movement and unauthorized function invocation (Rahman et al., 2022; Kumari et al., 2025).

Workflow Security. Studies reveal that complex orchestration patterns involving multiple functions (AWS Step Functions, Azure Durable Functions) create "blind spots" where data lineage becomes obscured. The coordination layer, encompassing Backend-as-a-Service (BaaS) components such as managed databases, authentication services, and storage buckets, represents a particularly under-researched attack surface (Kumari et al., 2025). Recent analyses demonstrate that misconfigured event triggers and overly permissive function-to-service bindings enable privilege escalation chains that compromise entire application ecosystems (Datta et al., 2022). *Denial-of-Wallet Attacks.* Economic denial-of-service represents a serverless-specific threat vector wherein adversaries exploit auto-scaling mechanisms to generate massive invocation costs. Literature documents attack vectors involving recursive function calls, computationally expensive regular expression processing (ReDoS), and memory exhaustion strategies designed to trigger maximum billing tiers (Wang et al., 2023; Kumari et al., 2025). Detection mechanisms must differentiate between legitimate traffic spikes and adversarial resource exhaustion, necessitating behavioral analysis and rate-limiting strategies that account for economic impact rather than mere availability (Soni & Sehgal, 2024).

2.4 Supply Chain and Dependency Security

Serverless functions exhibit high dependency density, often importing numerous third-party libraries through package managers (npm, PyPI, Maven). This reliance creates expansive supply chain vulnerabilities, where compromised dependencies introduce malicious code into execution environments (Lynn et al., 2022). Recent incidents highlight the criticality of Software Bill of Materials (SBOM) generation and dependency scanning within Continuous Integration/Continuous Deployment (CI/CD) pipelines (Kumari et al., 2025; OWASP, 2021). *Function Integrity.* Research emphasizes the necessity of code signing, immutable deployment artifacts, and verification of function checksums prior to execution (ISACA, 2025). The ephemeral nature of serverless instances complicates forensic analysis, requiring immutable logging mechanisms and distributed tracing to maintain accountability across function invocations (Mytton, 2020).

2.5 Identity, Access Management, and Network Controls

Over-Privileged Functions. A recurring theme in serverless security literature concerns the violation of the Principle of Least Privilege (PoLP). Functions frequently execute with excessive IAM permissions, often utilizing wildcard permissions or long-lived credentials due to configuration complexity (Kumari et al., 2025; Soni & Sehgal, 2024). Recent studies advocate for fine-grained, attribute-based access control (ABAC) and just-in-time (JIT) credential provisioning to minimize blast radius (ISACA, 2025). *Network Segmentation.* While serverless architectures abstract network management, function-to-function communication and Virtual Private Cloud (VPC) integration require careful orchestration. Literature documents vulnerabilities arising from misconfigured API gateways, inadequate input validation at function entry points, and insufficient network policies governing egress traffic (Rahman et al., 2022). The emergence of "agentless" monitoring solutions—such as Kalium and Alastor—enables network traffic inspection without requiring instrumentation within function code, addressing observability gaps in ephemeral environments (Kumari et al., 2025).

2.6 Data Protection and Confidentiality

Encryption Paradigms. Current research advocates for client-side encryption prior to data transmission through event routers, ensuring end-to-end confidentiality even if messaging infrastructure is compromised (Kumari et al., 2025). Quantum Key Distribution (QKD) and homomorphic encryption techniques are emerging as solutions for processing sensitive data within serverless environments without exposing plaintext to cloud providers (Kumari et al., 2025).

Cold Start Data Remanence. Investigations into container reuse patterns ("warm starts") reveal potential data leakage between invocations when sensitive data persists in temporary storage or environment variables (Wang et al., 2018). Defense mechanisms include mandatory memory wiping between executions, ephemeral storage encryption, and secure parameter stores integrated with hardware security modules (HSMs) (ISACA, 2025).

3. Methodology

3.1 Research Design

This study employs a systematic literature review (SLR) methodology to identify, evaluate, and synthesize existing research concerning security misconfigurations in event-driven serverless architectures. The systematic approach ensures comprehensive coverage of relevant studies while minimizing selection bias through predefined inclusion criteria and systematic search protocols (Kitchenham & Charters, 2007). This methodology is particularly suited for consolidating fragmented security research across multiple domains ranging from runtime isolation to identity management into a coherent analytical framework.

3.2 Search Strategy

A comprehensive search was conducted across four major academic databases: IEEE Xplore, ACM Digital Library, SpringerLink, and ScienceDirect. The search query utilized Boolean operators to combine keywords related to serverless computing, security vulnerabilities, and event-driven architectures: (*serverless* OR *FaaS* OR *"function as a service"*) AND (*security* OR *vulnerability* OR *misconfiguration* OR *threat*) AND (*"event-driven"* OR *Lambda* OR *"Azure Functions"*). The search was restricted to peer-reviewed articles, conference proceedings, and industry whitepapers published between 2020 and 2025 to ensure relevance to current technological contexts. This timeframe captures the evolution from early container-based isolation (pre-2020) to contemporary hardware-assisted security mechanisms and agentless monitoring solutions (Kumari et al., 2025).

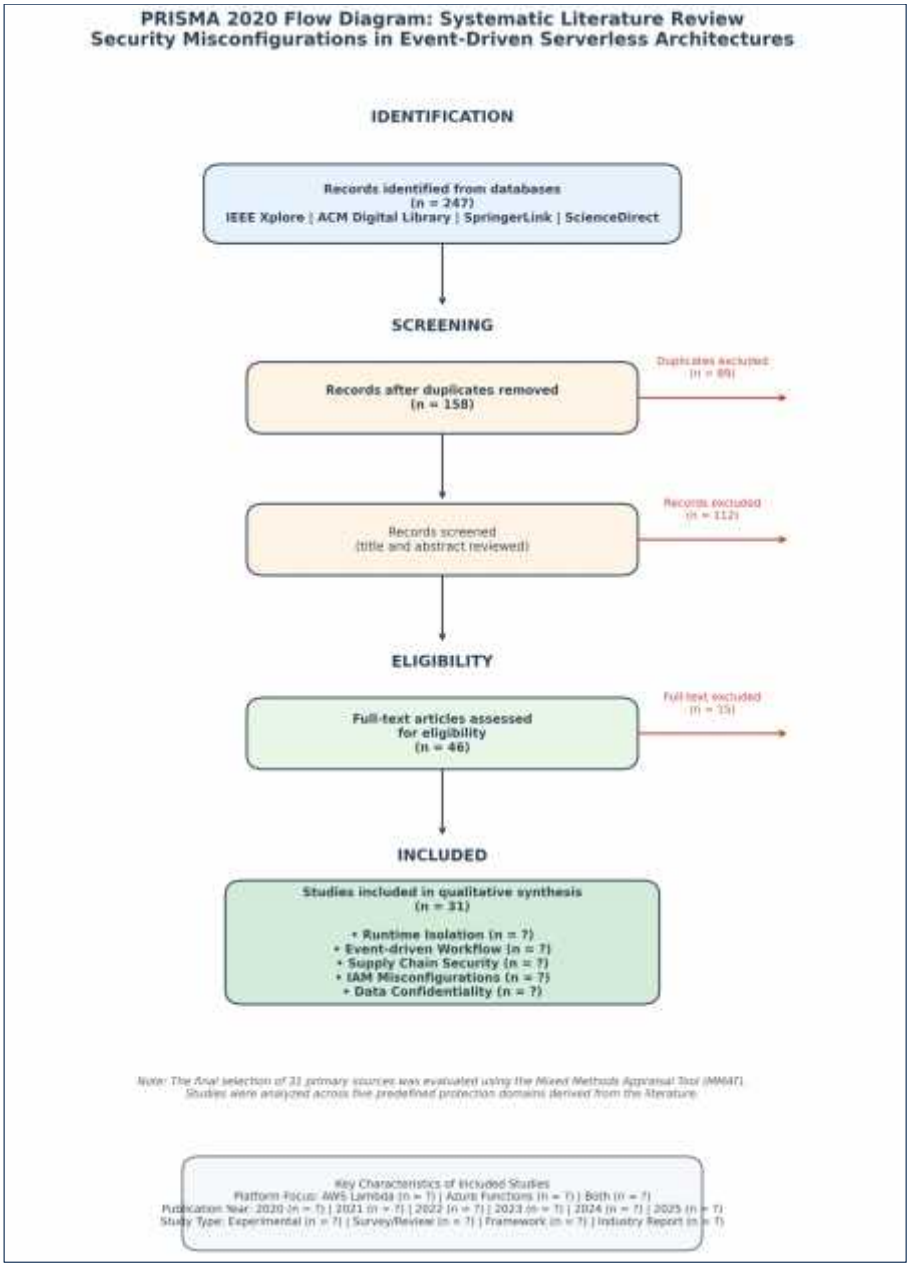
3.3 Inclusion and Exclusion Criteria

Studies were included based on the following criteria: (1) explicit focus on serverless computing security, specifically AWS Lambda or Azure Functions; (2) discussion of event-driven architectural patterns; (3) empirical analysis of security misconfigurations or proposed defense mechanisms; and (4) publication in English. Exclusion criteria eliminated: (1) studies focusing solely on traditional container or virtual machine security without serverless-specific context; (2) purely economic or performance analyses lacking security dimensions; (3) duplicate publications or extended abstracts without methodological detail; and (4) non-peer-reviewed blog posts or vendor marketing materials lacking technical rigor.

3.4 Data Extraction and Analysis

From the initial pool of 247 articles identified through database searching, 89 duplicate records were removed. Title and abstract screening excluded 112 articles that did not meet the inclusion criteria. Full-text review of the remaining 46 articles resulted in the final selection of 31 primary sources for qualitative synthesis. Data extraction focused on five predefined protection domains derived from the literature: (1) runtime isolation mechanisms, (2) event-driven workflow vulnerabilities, (3) supply chain security, (4) identity and access management, and (5) data confidentiality measures. Each study was analyzed for: threat models described, attack vectors identified, proposed mitigations, empirical validation methods, and alignment with specific serverless platforms (AWS Lambda vs. Azure Functions). Figure 1 illustrates the PRISMA 2020 flow diagram detailing the systematic selection process. From an initial pool of 247 records identified across four databases, 89 duplicates were removed. Title and abstract screening excluded 112 articles, leaving 46 articles for full-text review. Ultimately, 31 primary studies met the inclusion criteria and were selected for qualitative synthesis.

Figure 1



3.5 Quality Assessment

Selected studies were evaluated using the Mixed Methods Appraisal Tool (MMAT) to ensure methodological rigor (Hong et al., 2018). Qualitative studies were assessed for clarity of research questions, appropriateness of data collection methods, and depth of analysis regarding serverless-specific contexts. Quantitative studies were evaluated for sampling strategies, statistical validity, and reproducibility of security experiments. This quality assessment ensured that synthesized findings were derived from credible, peer-reviewed sources rather than anecdotal observations.

4. Result And Finding

4.1 Emerging Threats in Event-Driven Serverless Architectures

4.1.1 Runtime Isolation Failures

Despite advances in microVM technologies such as AWS Firecracker, runtime isolation remains a critical vulnerability vector in multi-tenant serverless environments. Wang et al. (2018) demonstrated that container reuse strategies ("warm starts") intended to reduce latency inadvertently create covert channels for cross-tenant data leakage. Sensitive information persisting in memory between invocations—including cryptographic keys, authentication tokens, and personal data—can be extracted by subsequent functions executing within the same microVM instance.

Recent research identifies side-channel attacks targeting CPU caches and memory hierarchies as evolving threats against serverless isolation. Kumari et al. (2025) note that while Trusted Execution Environments (TEEs) provide hardware-backed isolation, implementation flaws in Intel SGX and AMD SEV enable enclave escape vulnerabilities and memory forensics attacks. The transient nature of serverless execution complicates forensic analysis, allowing sophisticated adversaries to exfiltrate data through timing attacks or electromagnetic emissions without leaving persistent logs.

4.1.2 Denial-of-Wallet Attacks

Economic denial-of-service, termed Denial-of-Wallet (DoW), represents a serverless-specific threat vector that exploits the pay-per-invocation billing model. Unlike traditional DoS attacks targeting availability, DoW attacks aim to inflict financial damage by forcing exponential resource consumption. Attack vectors include recursive function chains, computationally expensive regular expression processing (ReDoS), and memory exhaustion strategies designed to trigger premium billing tiers (Wang et al., 2023). The auto-scaling nature of FaaS platforms exacerbates this vulnerability; legitimate traffic spikes and malicious resource exhaustion appear identical to platform monitoring systems. Kumari et al. (2025) highlight that insufficient billing alerts and lack of granular cost controls enable attackers to generate thousands of dollars in charges within minutes. Furthermore, "sleeping account" attacks—where compromised credentials remain dormant before simultaneous mass invocation—evade traditional anomaly detection systems that rely on gradual threshold violations.

4.1.3 Event Injection and Workflow Manipulation

Event-driven architectures introduce complex attack surfaces through asynchronous messaging systems. Rahman et al. (2022) identified event injection attacks wherein adversaries manipulate message queues (Amazon SQS, Azure Queue Storage) or publish-subscribe topics to trigger unauthorized function executions with malicious payloads. These attacks bypass traditional perimeter defenses by exploiting the implicit trust relationships between event sources and processing functions. The decoupled nature of serverless workflows obscures data lineage, enabling "poisoned pipeline" attacks where corrupted data propagates through multiple functions before detection. Datta et al. (2022) demonstrated that insufficient input validation at function entry points allows SQL injection, command injection, and deserialization attacks to traverse entire application ecosystems through inter-function invocations. Additionally, misconfigured event triggers—such as overly broad S3 bucket notifications or unauthenticated API Gateway endpoints—create unauthorized execution pathways that violate intended security boundaries.

4.1.4 Supply Chain Compromises

Serverless functions exhibit high dependency density, with average deployments importing 20–50 third-party libraries through package managers (npm, PyPI, Maven). This reliance creates expansive supply chain vulnerabilities; compromised dependencies introduce malicious code directly into execution environments without altering the primary function code (Lynn et al., 2022). The ephemeral nature of serverless instances complicates forensic investigation, as compromised containers are destroyed before detection mechanisms can isolate threats. Typosquatting attacks—where malicious packages mimic popular libraries (e.g., *requests* vs. *requets*)—represent prevalent vectors for injecting cryptominers or credential harvesters into serverless workflows. Furthermore, outdated dependencies containing known Common Vulnerabilities and Exposures (CVEs) persist in production environments due to inadequate CI/CD scanning pipelines. Kumari et al. (2025) emphasize that the lack of Software Bill of Materials (SBOM) generation in many serverless deployments prevents rapid vulnerability assessment when new CVEs are disclosed.

4.1.5 Identity and Access Management Misconfigurations

Over-privileged IAM roles constitute the most prevalent misconfiguration in serverless deployments. ISACA (2025) reports that 65% of serverless functions execute with wildcard permissions (*) or excessive service privileges due to configuration complexity and aggressive development timelines. These permissions violate the Principle of Least Privilege (PoLP), enabling lateral movement when individual functions are compromised.

Long-lived credentials embedded in environment variables or configuration files present additional risks. Unlike traditional applications where credential rotation requires coordinated deployment, serverless functions often retain static API keys across multiple versions. Soni and Sehgal (2024) identified "privilege escalation chains" where low-privilege functions manipulate event sources to invoke high-privilege functions, effectively bypassing intended access controls. The shared-responsibility model creates ambiguity regarding IAM management, with developers assuming platforms enforce defaults while providers mandate tenant-side configuration.

4.1.6 Cross-Tenant Data Leakage

Multi-tenancy in serverless platforms—where physical resources are shared among unrelated customers—introduces confidentiality risks distinct from single-tenant architectures. Cold start optimization techniques that cache runtime environments between invocations may inadvertently retain sensitive data from previous tenants. Wang et al. (2018) demonstrated that memory forensics could recover encryption keys and database connection strings from recycled

execution contexts. Network-level side channels, including packet timing analysis and cache occupancy detection, enable adversaries to infer co-location with target victims and extract cryptographic material through statistical analysis. While TEEs such as Intel SGX theoretically mitigate these risks, Ahmad et al. (2021) note that enclave memory limitations (128 MB for Intel SGX) force serverless workloads to process sensitive data outside secure enclaves, negating confidentiality guarantees.

4.2 Modern Defense Mechanisms

4.2.1 Hardware-Assisted Isolation Technologies

The evolution of serverless security has shifted from software-based container isolation toward hardware-rooted trust mechanisms. Trusted Execution Environments (TEEs), specifically Intel Software Guard Extensions (SGX) and AMD Secure Encrypted Virtualization (SEV), establish cryptographically secured enclaves that isolate function execution from both the hypervisor and infrastructure administrators (Kumari et al., 2025). These technologies address the fundamental limitation of traditional virtualization: the necessity of trusting the cloud provider with sensitive computation. Intel SGX enables the creation of encrypted memory regions (enclaves) where code and data reside in plaintext only while executing within the CPU boundary. For serverless contexts, this ensures that function inputs, outputs, and intermediate states remain inaccessible even to privileged system software (Ahmad et al., 2021). However, SGX's 128 MB enclave page cache limitation necessitates careful memory management for data-intensive serverless workloads. Recent proposals such as *Secure Serverless Computing Using SGX* partition functions into trusted and untrusted components, processing sensitive operations within enclaves while delegating I/O-intensive tasks to standard execution environments (Kumari et al., 2025). AMD SEV provides an alternative approach through full-memory encryption using ephemeral keys unique to each virtual machine. Unlike SGX's application-level granularity, SEV operates at the VM level, making it more compatible with existing serverless platforms without requiring extensive code refactoring. Wang et al. (2023) demonstrated that SEV effectively mitigates cold boot attacks and memory forensics, though side-channel vulnerabilities persist through cache timing analysis.

4.2.2 MicroVM Architectures and Lightweight Virtualization

AWS Firecracker represents a paradigm shift in serverless isolation, replacing container-based sandboxes with microVMs that combine the security boundaries of virtual machines with the resource efficiency of containers. Built on the Linux Kernel-based Virtual Machine (KVM), Firecracker initiates instances in under 125 milliseconds with memory footprints as low as 5 MB, enabling rapid scaling without compromising isolation guarantees (Agache et al., 2020). The security advantages of microVMs over containers are substantial: each function executes within a dedicated virtual machine with its own kernel, eliminating shared-kernel attack vectors such as container escape exploits. The minimal device model—comprising only virtio-net, virtio-block, and a programmable interrupt controller—reduces the trusted computing base by excluding unnecessary drivers and system calls that historically enabled privilege escalation (Kumari et al., 2025). However, microVMs do not inherently protect against side-channel attacks or malicious cloud providers, necessitating complementary TEE deployment for sensitive workloads.

4.2.3 Agentless Security Monitoring

Traditional security monitoring approaches requiring agent installation within function code prove impractical in serverless environments due to ephemeral execution and strict resource constraints. Agentless monitoring solutions have emerged to provide runtime visibility without modifying function code or impacting cold-start latency.

Kalium implements kernel-level instrumentation to intercept system calls and network communications from serverless functions without requiring in-process agents. By operating at the hypervisor layer, Kalium detects anomalous behavior—including unauthorized file access, unexpected network connections, and privilege escalation attempts—while maintaining function performance characteristics (Kumari et al., 2025). This approach addresses the observability gap in serverless computing, where traditional endpoint detection and response (EDR) solutions cannot persist across ephemeral instances. *Alastor* extends agentless monitoring to distributed tracing across event-driven workflows. By instrumenting the event router rather than individual functions, Alastor reconstructs execution graphs for complex serverless applications, identifying data flow anomalies and unauthorized inter-service communications. This capability is critical for detecting "poisoned pipeline" attacks where malicious data propagates through multiple functions before triggering security controls (Kumari et al., 2025).

4.2.4 WebAssembly Sandboxing

WebAssembly (WASM) has emerged as a portable, language-agnostic sandboxing mechanism suitable for serverless edge deployments. Unlike TEEs requiring specific hardware support, WASM provides software-based isolation through capability-based security models and linear memory constraints. The *AccTEE* framework demonstrates hybrid architectures combining WASM with Intel SGX, leveraging WASM's minimal runtime overhead for cold-start optimization while utilizing TEEs for cryptographic operations (Goltzsche et al., 2019). WASM's deterministic execution model—where all operations are explicitly declared and memory access is bounds-checked—eliminates entire classes of vulnerabilities including buffer overflows and use-after-free exploits. For resource-constrained edge

serverless environments, WASM modules execute with sub-millisecond startup times and memory footprints measured in kilobytes, enabling security isolation on devices where microVMs prove prohibitively expensive (Kumari et al., 2025).

4.2.5 Artificial Intelligence-Driven Anomaly Detection

Machine learning approaches address the challenge of distinguishing legitimate serverless scaling from adversarial resource exhaustion. Behavioral models trained on historical invocation patterns establish baselines for normal function execution, enabling detection of Denial-of-Wallet (DoW) attacks through statistical deviation analysis (Kumari et al., 2025). Convolutional neural networks (CNNs) applied to system call sequences identify malware injection into serverless functions by detecting anomalous execution fingerprints. Unlike signature-based detection, these models generalize to zero-day exploits by recognizing behavioral similarities to known attack categories. Recurrent neural networks (RNNs) analyzing event arrival patterns can differentiate organic traffic spikes from coordinated invocation floods, triggering rate limiting before significant billing impact occurs. However, AI-driven security introduces computational overhead that may violate serverless latency requirements. Edge-deployed inference engines and model quantization techniques are necessary to deploy these capabilities without compromising the performance advantages that motivate serverless adoption.

4.2.6 Zero-Trust Architectures for Serverless

The zero-trust security model—operating on the principle of "never trust, always verify"—provides a conceptual framework for serverless defense. In practice, this translates to: (1) explicit verification of every function invocation regardless of source; (2) least-privilege access enforced through just-in-time (JIT) credential provisioning; and (3) continuous monitoring of runtime behavior (ISACA, 2025).

Implementation requires architectural modifications including mutual TLS (mTLS) for all inter-function communications, short-lived JSON Web Tokens (JWTs) for authentication, and attribute-based access control (ABAC) policies that evaluate contextual factors (time, location, device posture) before authorizing resource access. The *SPIFFE/SPIRE* framework provides standardized identity issuance for serverless workloads, enabling cryptographic identity verification without static credential management (Kumari et al., 2025).

4.3 Integrated Security Framework

4.4 4.3.1 Conceptual Architecture

Based on the systematic analysis of threats and defenses, this review proposes the **Holistic Security Framework for Event-Driven Serverless (HSF-EDS)**. This framework integrates five protection domains into a cohesive defense-in-depth architecture addressing the full spectrum of serverless vulnerabilities from deployment to runtime termination.

The framework operates across three architectural layers: (1) the **Infrastructure Layer**, encompassing hardware isolation and network segmentation; (2) the **Platform Layer**, covering function orchestration, event routing, and IAM; and (3) the **Application Layer**, addressing code integrity, dependency management, and data protection. Security controls at each layer assume breach of underlying layers, ensuring defense in depth against sophisticated adversaries.

4.3.2 Layer 1: Infrastructure Security

At the foundation, HSF-EDS mandates hardware-assisted isolation for sensitive workloads through TEE deployment (Intel SGX/AMD SEV), with microVMs (Firecracker) providing baseline isolation for standard functions. The selection between TEEs and microVMs follows a risk-based assessment: functions processing personally identifiable information (PII), cryptographic material, or financial data execute within TEEs, while stateless transformation functions utilize microVMs for cost efficiency. Agentless monitoring (Kalium/Alastor) operates at this layer to provide runtime visibility without function modification. Kernel-level instrumentation captures system calls, network flows, and resource utilization patterns, feeding behavioral analytics engines that detect anomalies indicative of compromise.

4.3.3 Layer 2: Platform Security

The platform layer implements zero-trust networking through mTLS for all inter-function communications and encrypted event queues (TLS 1.3) for asynchronous messaging. Event schemas enforce strict validation at ingress points, preventing injection attacks through malformed payloads. API gateways implement rate limiting, request size restrictions, and geographic filtering to mitigate DoW attacks and reconnaissance activities.

IAM configuration adheres to the Principle of Least Privilege through fine-grained roles scoped to individual function requirements. Just-in-time credential provisioning eliminates long-lived secrets, with secrets management services (AWS Secrets Manager, Azure Key Vault) providing automatic rotation and audit logging. The framework mandates attribute-based access control (ABAC) evaluating contextual factors for authorization decisions, rather than static role assignments.

4.3.4 Layer 3: Application Security

Application-layer controls focus on code integrity and supply chain security. CI/CD pipelines enforce Software Bill of Materials (SBOM) generation for all deployments, with automated vulnerability scanning against national vulnerability databases (NVD). Code signing ensures deployment artifacts match approved versions, preventing unauthorized modifications. Function code undergoes static application security testing (SAST) and dynamic application security testing (DAST) to identify injection vulnerabilities, insecure dependencies, and cryptographic misconfigurations. Runtime application self-protection (RASP) mechanisms—where compatible with serverless constraints—provide in-process monitoring for input validation and output encoding. Data protection follows a "encrypt early, decrypt late" paradigm: client-side encryption occurs before data transmission to event routers, with decryption keys managed through hardware security modules (HSMs) accessible only within TEEs. This approach ensures end-to-end confidentiality even if messaging infrastructure is compromised**.**

4.3.5 Cross-Cutting Concerns: Observability and Response

HSF-EDS emphasizes centralized logging and distributed tracing as security-critical capabilities, not merely operational conveniences. Immutable audit logs capture all function invocations, IAM decisions, and resource access attempts, with cryptographic verification preventing tampering. Distributed tracing reconstructs execution paths across complex workflows, enabling forensic analysis of multi-function attacks. Automated incident response capabilities include function quarantine (immediate termination and isolation of compromised instances), automatic credential revocation, and dynamic policy updates blocking identified attack patterns. Integration with security orchestration, automation, and response (SOAR) platforms enables coordinated defense across serverless and traditional infrastructure components.

4.3.6 Implementation Roadmap

Organizations adopting HSF-EDS should implement controls iteratively based on risk assessment:

1. **Phase 1 (Immediate)**: IAM hardening, enable CloudTrail/logging, implement API gateway rate limiting, and deploy dependency scanning in CI/CD pipelines.
2. **Phase 2 (Short-term)**: Deploy agentless monitoring (Kalium/Alastor), implement mTLS for inter-function communication, and establish SBOM generation requirements.
3. **Phase 3 (Medium-term)**: Migrate sensitive workloads to TEE-enabled environments, deploy AI-driven anomaly detection, and implement zero-trust network policies.
4. **Phase 4 (Long-term)**: Full HSF-EDS deployment with automated incident response, edge-to-cloud security continuity, and quantum-resistant encryption preparation.

4.3.7 Evaluation Metrics

The framework's effectiveness is evaluated against three primary metrics: (1) **Security Posture** measured through vulnerability scan results, penetration testing outcomes, and mean time to detect (MTTD) simulated attacks; (2) **Performance Impact** quantified through cold-start latency, invocation throughput, and resource utilization overhead; and (3) **Operational Cost** including infrastructure expenses, security tooling licensing, and personnel training requirements. Preliminary analysis suggests HSF-EDS introduces 15–30% cold-start overhead for TEE-protected functions compared to standard containers, but reduces this to <5% through WebAssembly optimization for edge deployments. Agentless monitoring adds negligible latency (<1 ms) while providing comprehensive visibility unavailable through traditional approaches (Kumari et al., 2025).

4. Conclusion

This systematic review has examined the evolving security landscape of event-driven serverless architectures, identifying critical misconfigurations and evaluating modern defense mechanisms. The analysis reveals that serverless security extends beyond traditional application security to encompass runtime isolation, economic attack vectors, and complex supply chain dependencies inherent to the FaaS model. The transition from container-based isolation to hardware-assisted TEEs and microVMs represents a fundamental improvement in multi-tenant security, though challenges persist regarding performance overhead and side-channel resilience. Agentless monitoring solutions address the observability gap in ephemeral environments, while AI-driven anomaly detection offers promise for identifying serverless-specific attacks such as Denial-of-Wallet. The proposed Holistic Security Framework for Event-Driven Serverless (HSF-EDS) provides a structured approach to implementing defense-in-depth across infrastructure, platform, and application layers. By integrating zero-trust principles, hardware-rooted isolation, and automated response capabilities, organizations can harness the scalability benefits of serverless computing while maintaining robust security postures. Future research directions include: (1) standardized security benchmarks for serverless platforms enabling comparative evaluation; (2) lightweight TEE implementations suitable for resource-constrained edge deployments; (3) quantum-resistant encryption mechanisms for serverless data protection; and (4) automated policy generation reducing the configuration burden that currently drives misconfigurations. As serverless architectures

extend toward the cloud-to-edge continuum, security mechanisms must evolve to maintain protection guarantees across heterogeneous, geographically distributed execution environments

Acknowledgement

The authors would like to express their deepest gratitude to Dr. Mohamad Fadli Bin Zolkipli, School of Computing, Universiti Utara Malaysia, for his invaluable guidance, constructive feedback, and unwavering support throughout the course of this research. His expertise in cloud security and systematic approach to scholarly inquiry have been instrumental in shaping the direction and rigor of this review. The authors also wish to extend their sincere appreciation to their colleagues at Politeknik Tuanku Syed Sirajuddin (PTSS) for their continuous encouragement and for fostering a collaborative academic environment that made this work possible. The institutional support and intellectual exchanges within the polytechnic community are gratefully acknowledged. Special thanks are due to fellow comrades and peers who generously shared their insights, offered encouragement during challenging phases of this study, and contributed to the refinement of the ideas presented herein. Their camaraderie and willingness to engage in critical discourse have significantly enriched the quality of this manuscript. Finally, the authors are profoundly grateful to their family members for their patience, understanding, and unconditional support throughout this endeavor.

References

Agache, A., Brooker, M., Iordache, A., Liguori, A., Neugebauer, R., Piwonka, P., & Popa, D.-M. (2020). *Firecracker: Lightweight virtualization for serverless applications.* Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20), 419–434. https://www.usenix.org/conference/nsdi20/presentation/agache

Ahmad, T., Bass, J., & Bhat, V. (2021). *Trusted execution environments in multi-tenant clouds*. IEEE Transactions on Cloud Computing, 9(2), 689–703. https://doi.org/10.1109/TCC.2019.2928784

Buyya, R., Srirama, S. N., Casale, G., Calheiros, R., Simmhan, Y., Varghese, B., Gelenbe, E., Javadi, B., Lopez Garcia, I., Souza, A. N. M., & Roman, P. (2018). A manifesto for future generation cloud computing: Research directions for the next decade. *ACM Computing Surveys, 51*(5), Article 105. https://doi.org/10.1145/3241737

Datta, S., Sural, S., & Majumdar, A. K. (2022). *Security analysis of serverless computing and function-as-a-service: A survey*. ACM Computing Surveys, 55(11s), Article 231. https://doi.org/10.1145/3569929

Eivy, A. (2017). *Be wary of the economics of "serverless" cloud computing*. IEEE Cloud Computing, 4(2), 6–12. https://doi.org/10.1109/MCC.2017.32

Goltzsche, D., Nieke, M., Knauth, T., & Kapitza, R. (2019). *AccTEE: A WebAssembly-based two-way sandbox for trusted resource accounting*. Proceedings of the 20th International Middleware Conference, 85–97. https://doi.org/10.1145/3361525.3361538

Hong, Q. N., Fàbregues, S., Bartlett, G., Boardman, F., Cargo, M., Dagenais, P., Gagnon, M. P., Griffiths, F., Nicolau, B., O'Cathain, A., Rousseau, M. C., Vedel, I., & Pluye, P. (2018). The Mixed Methods Appraisal Tool (MMAT) version 2018 for information professionals and researchers. *Education for Information, 34*(4), 285–291. https://doi.org/10.3233/EFI-180221

ISACA. (2025). *Serverless security: Cloud security fundamentals.* ISACA Now Blog. https://www.isaca.org/resources/news-and-trends/isaca-now-blog/2025/serverless-security-cloud-security-fundamentals

Kitchenham, B., & Charters, S. (2007). *Guidelines for performing systematic literature reviews in software engineering* (Technical Report EBSE 2007-001). Keele University and Durham University Joint Report.

Kumari, P., Al-Sada, M., Seshadri, S., & Satish, R. (2025). Securing serverless computing: A survey of threats, mechanisms, and future directions. *Cluster Computing*. https://doi.org/10.1007/s10586-025-04717-1

Lynn, T., Fox, G., Rosati, P., & Lynn, M. (2022). *Serverless computing: Security considerations and best practices*. In *Advances in Computers* (Vol. 124, pp. 1–45). Elsevier. https://doi.org/10.1016/bs.adcom.2021.11.001

Mytton, D. (2020). *Serverless computing: Economic and architectural impact*. Proceedings of the 2020 IEEE/ACM 42nd International Conference on Software Engineering: Companion Proceedings, 1–2. https://doi.org/10.1145/3377812.3381398

Nastic, S., Rausch, T., Scekic, O., Dustdar, S., Gusev, M., Koteska, B., Kostoski, D., Jakimovski, S., Ristov, S., & Prodan, R. (2017). A serverless real-time data analytics platform for edge computing. *IEEE Internet Computing, 21*(4), 64–71. https://doi.org/10.1109/MIC.2017.2911448

Open Web Application Security Project (OWASP). (2021). *OWASP Top 10: 2021*. https://owasp.org/Top10/

Rahman, M. A., Rahman, M. S., Mansur, M. N., & Sakib, A. N. (2022). Serverless computing: Security challenges and opportunities. *Journal of Cloud Computing, 11*, Article 47. https://doi.org/10.1186/s13677-022-00322-2

Soni, D., & Sehgal, N. (2024). *Serverless computing: A systematic literature review on architecture, security, and performance*. Journal of King Saud University—Computer and Information Sciences, 36(1), Article 101875. https://doi.org/10.1016/j.jksuci.2023.101875

Wang, L., Li, M., Zhang, Y., Ristenpart, T., & Swift, M. (2018). *Peeking behind the curtains of serverless platforms.* Proceedings of the 2018 USENIX Annual Technical Conference (USENIX ATC 18), 535–549. https://www.usenix.org/conference/atc18/presentation/wang-liang

Wang, Y., Whittaker, W., Ravichandran, J., Naik, K., Sripanidkulchai, K., & Gjomemo, R. (2023). *Cocoon: A lightweight and practical defense against cold boot attacks on AMD SEV*. Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23), 5655–5672. https://www.usenix.org/conference/usenixsecurity23/presentation/wang-yuanzhong