International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246.IJETCSIT-V6I1P114
Eureka Vision Publication | Volume 6, Issue 1, 134-144, 2025

Original Article

Multi-Layered Security Policy Enforcement for Confidential Data in Serverless Cloud Functions

Srinivas Potluri
Director EGS Global Services

Abstract - Serverless computing has disrupted the way clouds bring customized applications to the market because it removes the need to manage the infrastructure, allowing flexible, event-based implementations. Nevertheless, this paradigm presents new security issues, particularly, how to manage and secure confidential information. Serverless functions have ephemeral, stateless, and distributed characteristics, exposing them to an elevated level of attack surfaces, misconfigurations, and privilege escalation threats. The paper offers a complex stacked policy enforcement model that protects sensitive data in the Function-as-a-Service (FaaS) computing environment, including AWS Lambda, Azure Functions, and Google Cloud Functions. The framework suggested contains five layers interconnected with each other: authentication and access control, data classification and isolation, a context-aware policy engine, runtime tracking with anomaly detection, and audit logging with compliance verification. In deployments to production and simulated attacks (Denial-Of-Service (DoS) and API injections, and data exfiltration), we show that our system can provide high mitigation rates (up to 99.1%) at low overhead (~11.2%). We also seek to apply reinforcement learning to dynamically update the policy and fit well into DevSecOp pipelines to partake in continuous protection. We also compare the performance of cold/warm starting, cross-cloud compatibility and evolution of policy over a long time. The findings emphasize the level of protection that the layered defence offers against serverless-based applications, and they also present the potential of automated policy synthesis and edge-cloud policy extension. This work provides a scalable and extensible future of safe policy-driven serverless computing.

Keywords - Serverless Computing, Confidential Data, Multi-Layered Security, Policy Enforcement, AWS Lambda, Azure Functions, Function-as-a-Service.

1. Introduction

Serverless computing has been spreading rapidly and profoundly changing the cloud computing landscape, as it hides infrastructure management and enables developers to focus solely on application logic. Serverless platforms, including AWS Lambda, Google Cloud Functions, and Microsoft Azure Functions, automatically scale, are cost-efficient, and have much shorter deployment cycles, realized through the use of cloud provider-managed environments. [1-3] This abstraction, however, also includes costs in terms of visibility and control, which critically raises concerns about the security of data when it comes to handling confidential or sensitive information.

Serverless functions are ephemeral, stateless, and, often, short-lived; that is, unlike traditional cloud models, where applications execute in clearly defined virtual machines or containers. Such attributes make it challenging to implement sustainable security policies in diverse execution scenarios. A lack of isolation for confidential data, an implicit assumption of trust, or inadequate access control mechanisms may expose confidential data to unauthorised access, leakage, or misuse. Moreover, being event-driven and often connecting to multiple other cloud services, the serverless function environment is more dynamic and multifaceted, requiring a more mature and flexible security model.

Classic security technology is insufficient in the context of serverless architecture, as it is commonly designed to support long-lived stateful environments. The result is that speciality security structures are increasingly demanded, which address the specific properties of serverless computing. This study presents a multi-authored security policy enforcement architecture that focuses on securing confidential data in serverless clouds. The framework enforces security policies throughout the lifecycle of functions to be deployed, executed, and beyond, integrating static analysis, context-aware dynamic monitoring, protection through fine-grained access controls, and data encryption.

The suggested solution is platform-independent and scalable, allowing it to seamlessly integrate into open serverless workflows without a noticeable performance impact. The system actively assists in identifying and resolving policy violations, enhancing confidence in serverless applications, and ensuring compliance with data protection regulations. In this paper, the design, realisation, and testing of the described framework are discussed, which provides a realistic and secure way forward for organisations that need to leverage serverless technology while meeting certain confidentiality requirements.

2. Background and Related Work

Serverless computing can be considered a revolution in cloud-native application development. Serverless enables the abstraction of the underlying infrastructure, allowing developers to deploy individual functions that automatically scale with demand. Although this model is more agile and cost-effective, it also introduces new security issues that need to be addressed, particularly when handling proprietary data. This section provides an overview of the serverless computing paradigm, security issues associated with these architectures, and current methodologies for policy implementation, with an emphasis on securing sensitive data.

2.1. Serverless Computing Models

Serverless applications are built on the foundation of Function-as-a-Service (FaaS), where developers implement stateless functions that are executed in response to specific events, such as an HTTP request, file upload, or database update. [4-6] The functions run inside ephemeral containers that are managed by and provisioned by cloud providers. This elastic and dynamic aspect of FaaS means that serverless applications can scale within seconds, and scaling does not require any configuration or interaction from the developer. This, in turn, permits transparent, fine-grained billing at the actual time the compute resources are utilised.

Regardless of its efficiency, the serverless model has peculiarities in its operations. Functions are typically executed over short intervals and are independent of one another. Although this is one of the benefits of microservices and a decoupled application design, it introduces complexity to issues such as function orchestration and managing long-lived state. Additionally, the virtualisation of servers hinders visibility into the time environment, discouraging the use of traditional security monitoring and control measures. Even though the concept of serverless implies that one no longer needs servers, some form of infrastructure is still involved; it simply is not apparent to the end user.

2.2. Security Challenges in Serverless Architectures

Serverless systems have several security concerns that differ from those observed in traditional server-based systems. The increased attack surface area is one of the biggest issues because functions can be initiated by a variety of sources, including APIs, storage triggers, or Internet of Things devices. This diversity predisposes to more exposure to injection attacks, payload malformation, etc. Another major problem is the weakness of misconfiguration. The parameters of the functionality with poorly defined timeouts or concurrency may result in Denial-of-Service (DoS) or Denial-of-Wallet (DoW)-based attacks. The most common attacks against the serverless model in such cases will be cost usury or the function capacity overflow due to the pay-per-use nature.

A stateless and distributed microservices-based system may experience access control vulnerability due to an authentication weakness. The functions frequently run in isolation, and the failure of one function provides attackers with an opportunity to exploit related services or gain higher privileges. Finally, there exist great dangers associated with over-privileged functions. An overuse of privileges in one functionality (e.g. privileged access to a database) can enable an attacker to perform lateral movement in a way that conflicts with the principle of least privilege. These problems are further exacerbated by cold-start latency, third-party dependencies, and the limitations of the serverless environment in terms of debugging.

2.3. Existing Policy Enforcement Techniques

In response to these challenges, several mechanisms for policy enforcement have been designed at both the infrastructure and application levels. Controls at the infrastructure level: These controls continuously examine and scan configuration settings, seeking potentially dangerous vulnerabilities such as publicly exposed secrets or excessively broad roles. Cloud Workload Protection Platforms (CWPPs) provide runtime protection mechanisms that monitor function execution to detect anomalies, such as unexpected outbound traffic or suspicious Application Programming Interface (API) calls. Such systems can monitor the exfiltration of data or tampering with functions in real-time.

Also, proactive code instrumentation has been identified as a good approach. Securities policies are coded directly into the function codebase, allowing functions to validate their inputs, perform authorisation checks, and enforce regulatory compliance before execution. With the assistance of this so-called shift-left, security is embedded into the growth process at an earlier stage, allowing for the identification of vulnerabilities before launch. These solutions focus on enforcing least-privilege access, validating data flow, and securing deployment configurations. Their efficacy, however, depends on the serverless platform and the breadth of security controls the cloud provider provides.

2.4. Confidential Data Protection in Cloud Environments

The security of confidential information in serverless requires an overall protocol involving encryption, secret management, and policy-based access restriction. The bulk of cloud vendors offer some form of managed secret service (e.g., AWS Secrets Manager, Azure Key Vault) to securely store and retrieve API keys, credentials, and tokens, thereby preventing them from being hardcoded in application code. Data-centric policies are also essential. Function fusion techniques, where functions are merged to diminish inter-function exposure to such data, and intelligent scheduling, where schedules are run to bound the existence and extent to which data is exposed in the serverless workflow, are techniques intended to restrict the lifetime and extent to which data is exposed in the serverless workflow. These methods also reduce the chance of data leakage by minimising data persistence and making functions independent.

Cloud Infrastructure Entitlement Management (CIEM) and Data Security Posture Management (DSPM) are gaining popularity to promote visibility. CIEM aims to ensure that access controls over serverless features and functions adhere to the principle of least privilege. In the meantime, the DSPM platforms can visualise the data flows that include sensitive data within the application landscape and enable recognition of where confidential data is processed and consumed. Irrespective of such developments, problems persist. The lack of uniform policy enforcement across hybrid or multi-cloud environments is made challenging by vendor lock-in, state retention, limited functionality, and the transient nature of serverless functions. This sophistication explains why multi-layered, flexible security models will be necessary to provide comprehensive protection of confidential data in serverless environments.

3. System Architecture and Threat Model

3.1. Overview of the Proposed Framework

The targeted multi-layered security policy enactment architecture aims to ensure the security of confidential information in serverless cloud scenarios by combining coordinated units deployed at deployment, runtime, and enforcement of security-related policies. [7-10] The framework also includes stakeholders as DevOps engineers, security administrators, cloud providers, and end users, who collaborate in the form of a structured pipeline that integrates security into the lifecycle of the functions in a much deeper manner. The infrastructure itself (such as AWS, Azure, or GCP) used by the cloud provider forms the core of the system, and it provides APIs to deploy functions, control policies, and store sensitive information. Developers can upload serverless code with metadata and policies directly using the API function deployment, and security administrators can specify global security policies using a policy management console. Such policies are transferred to the enforcement framework, where the Policy Definition and Distribution component transforms them and communicates to the appropriate enforcement modules.

When policies are in place, the framework becomes active and its Data Classification and Isolation engine comes into action, examining data flows to mark information with a sensitivity rating (e.g., PII, financial books). This directly feeds into the Identity and Access Control Layer, which implements strict role-based access rules through token-based validation mechanisms combined with cloud IAM and Key Vault services. Both identity and context are validated in every function invocation, ensuring that only the correct conditions and users are allowed to access functions.

The Monitoring and Anomaly Detection component takes the security aspect into account at runtime, observing the behaviour of functions, data I/O, and events during execution in an active manner. It sends alarms in the event of suspicious activities, such as accessing rates higher or lower than normal, or unauthorised access. These alerts are filtered through the Context-Aware Policy Engine, which applies fine-grained security policies dependent on runtime variables such as the time of day, location, or even role. This dynamic enforcement plays an important role in the execution of ephemeral functions common in serverless models. Lastly, all interactions are recorded and sent to the Audit Logging and Compliance Validation system. This module generates a report after each execution for administrators, verifying that all security policies were enforced and allowing for forensics in the event of an incident. It also contributes to the policy definition module, creating a loop in security. The ability to enforce all these layers within its framework not only promotes a proactive defence but also reactive auditing and compliance that monitors the confidentiality of sensitive data in dynamic cloud environments.

3.2. Operational Assumptions and Environmental Scope

The proposed multi-layered security enforcement framework is based on the following operational assumptions and a well-defined scope of the environment in which the framework will be implemented. [11-13] To begin with, it is presumed that the base level of security is provided by the underlying Cloud Service Provider (CSP) like AWS, Azure, or GCP that implies hardening infrastructure, securely using virtualization, and certifying against the international set of standards (e.g., ISO 27001, SOC 2). Infrastructure and serverless execution environment security are thus delegated to the domain of the cloud provider, allowing the security framework to concentrate its efforts primarily on the application (or, rather, function) level and related controls. Additionally, it is assumed that any serverless functions will be deployed to a controlled cloud region where the cloud provider implements its Function-as-a-Service (FaaS) runtime execution. Such functions are event-driven (predefined events cause the functions to run) but do not maintain any state, and they will operate in ephemeral containers. Confidential information compiled by these functions is locally cached by the native cloud services (as object or managed database storage), referenced through IAM roles, and accessed via a secure API gateway. Secure key management services and token-based identity validation facilities are also utilised in the operational context.

Multi-function and serverless applications have a limited scope within the environment, including external services, internal APIs, and cloud storage. The framework does not control how cloud-native services operate internally (e.g., database engines). However, it does regulate the exchange of sensitive data between services, as well as between services and users, and between services and serverless functions. Notably, the model presupposes that the DevOps and security teams are tasked with defining policies and classifying information before it is deployed, and runtime checks are performed by automated enforcement tools integrated into the architecture.

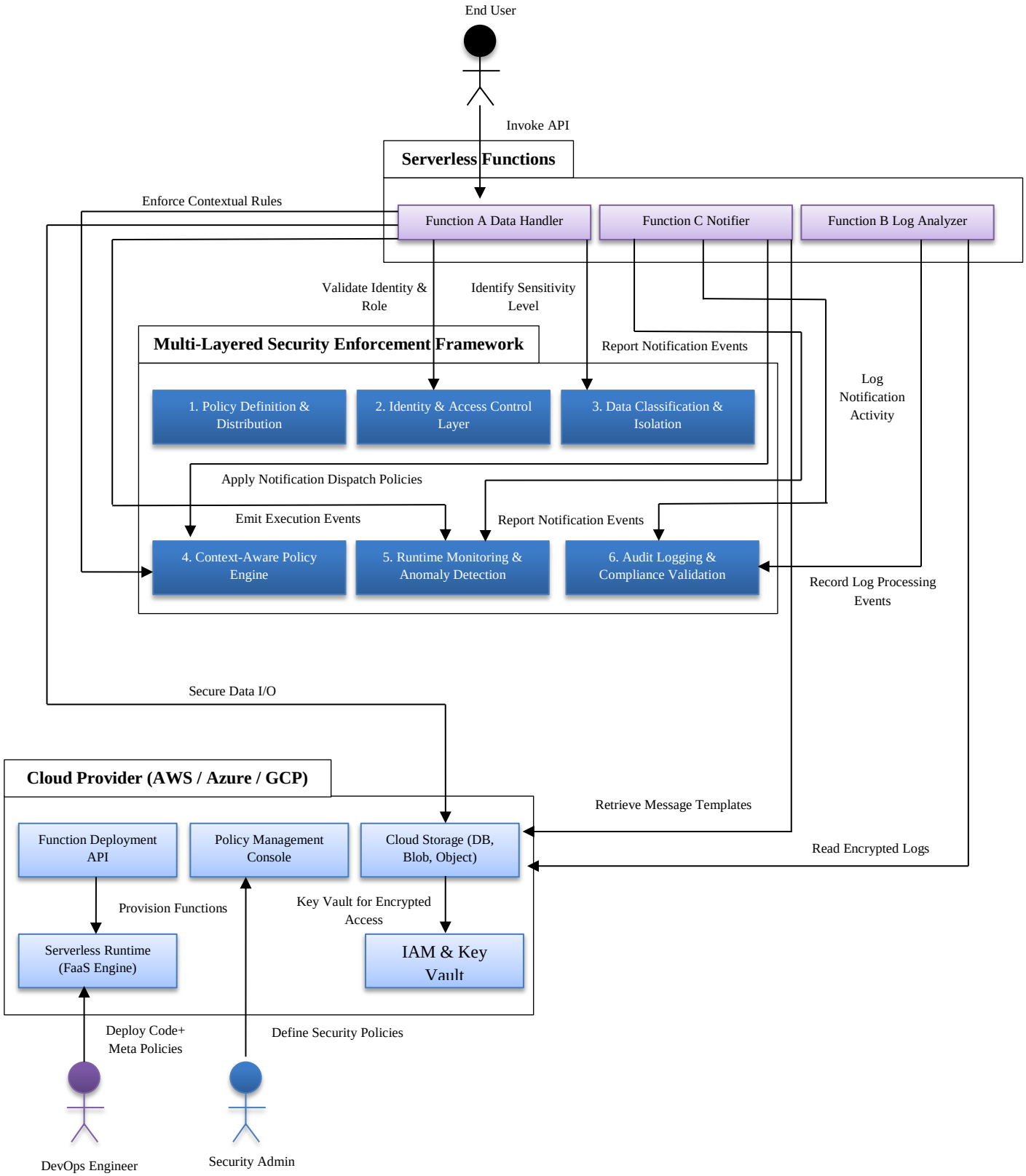

Figure 1. Multi-Layered Security Policy Enforcement Framework for Confidential Data in Serverless Cloud Functions

3.3. Threat Landscape and Adversary Capabilities

The increase in the size of the attack surface in serverless systems is primarily caused by the loose coupling between microservices and the fact that multiple events can be used to run one or more functions in a serverless environment. The framework is also expected to handle a broad palette of attackers, including opportunistic individuals as well as particularly sturdy and financially capable adversaries with more complex and sophisticated capabilities. The threats include unauthorised data access, code injection by injecting event payloads, gaining privileges by relying on misconfigured access, and both resource starvation and high costs due to denial-of-service attacks.

Specifically, one of the most troublesome risks is the exploitation of overprivileged functions, where a compromised function gains access to a wide range of cloud resources with overly permissive IAM roles. Hackers can navigate sideways through the system using excess privileges to completely steal information or continue attacking the application. Moreover, attackers can exploit improper configuration gaps, such as a generous time limit or wide API gates, to initiate prolonged attacks or provoke financial drain through Denial-of-Wallet (DoW) attacks.

The adversary model presupposes that attackers can be aware of cloud-native patterns and attempt to evade security by either direct (via invoking the function) or indirect (manipulating metadata or launching attacks based on replay) methods. Although it is possible that some adversaries could be external threat agents, the model also takes into account insider threat individuals who have legitimate access but may abuse their rights to gain unauthorised access to sensitive information. The combination of continuous monitoring, real-time access validation and context-aware enforcement components into a proposed framework works against these threats, as even momentary attacks can be tracked and brought under control.

3.4. Real-World Application Context: A Serverless Use Case

To ensure the security and privacy mechanisms proposed in the study translate to the real world, one might consider the following use case scenario: a healthcare app implemented using a serverless-based architecture that works with patient data and diagnostic files. In this case, two operations are carried out on file intake, including image processing with machine learning and safe notification to doctors, as well as audit logging to satisfy regulatory requirements. A set of storage events (e.g., a new file upload), HTTP API calls (e.g., a doctor accessing records), as well as message queues (e.g., task processing pipelines) trigger these functions.

Controlled healthcare information should be protected behind robust confidentiality measures due to its high sensitivity and laws such as HIPAA. The ingestion function, already vetted via identity and access controls, automatically classifies the file as sensitive based on metadata and content type once it is uploaded. Such classification initiates policies to restrict downstream access, permitting identified useful functions or roles (e.g., certified radiologists) to work with or observe the data. The context-aware policy engine reviews every access request using the context-based metadata of each request to contextualise it (e.g., time, location, and purpose).

Suppose runtime monitoring mechanisms detect an anomaly, such as an access request in an unusual location or an unexpected increase in API usage. In that case, they will flag the occurrence, log it as an audit marker, and, optionally, pause the execution process. The logs are transferred to the compliance validation system to verify that organisational and legal policies were not breached during data access. This use case again demonstrates that the granular, real-time enforcement that is made possible by the proposed multi-layered framework does not come at the expense of the agility and scalability advantages of a serverless framework.

4. Multi-Layered Security Policy Enforcement Mechanism

The framework represents a multi-layered enforcement model addressing the security complexities associated with confidential data in serverless environments. [14-17] All the layers are pre-configured to create a unique line of defence so that access to sensitive data is authorized only within the nationally formulated and verifiable conditions. Such a layered architecture will increase resilience by leveraging identity validation, data sensitivity tagging, and situational awareness, thereby preventing a wide range of risks, including unauthorised access and contextual misuse.

4.1. Layer 1: Authentication and Access Control

The initial and lower tier is authentication and access control, and only legitimate users or services should be able to invoke functions or access data. Identity and Access Management (IAM) systems provided by the cloud platform are used to perform identity verifications and implement either a token-based or certificate-based authentication system.

For human users, authentication is performed through integration with identity providers (e.g., Azure Active Directory, AWS IAM, or OAuth-based SSO integrations). For machine identities, there are service roles and policies that perform access control on a fine-grained level. The access control policy is based on the principle of least privilege, which grants users and functions the minimal base permissions necessary. Any serverless role is checked against a policy database before executing any serverless function to ensure it falls within the permitted access context. Additionally, within this framework, there are fine-grained controls (e.g., attribute-based access control: ABAC), which respond to requests by considering user attributes, resource tags, and environmental conditions (e.g., location, time of day). All these security

measures provide initial protection against impersonation attacks, privilege elevation, and unauthorised invocation of functions.

4.2. Layer 2: Data Classification and Isolation

The second level involves categorising data by level of sensitivity and implementing isolation for various types of data. When data are ingested or created, they are automatically labelled with a classification label: Public, Internal, Confidential, or Highly Confidential, depending on content analysis, metadata, or external instructions. These tags are essential to downstream enforcement, as they denote which operations can and cannot be conducted by whom.

To prevent data tampering or unauthorised data mixing, the framework ensures isolation at both the storage level and in available functions. As an illustration, documentation could be stored in encrypted buckets with limited access keys; official information could be made available through broader scopes. The risk of cross-contamination is reduced by using logically separated environments or virtual networks to deploy functions that process different classes of data. When a particular function attempts to access data, its purpose and intended execution are compared to the classification of the data, and it must satisfy the predetermined policy levels. This layer reduces the risks of data exfiltration and enhances regulatory compliance, particularly in settings that process personally identifiable information (PII) or financial data.

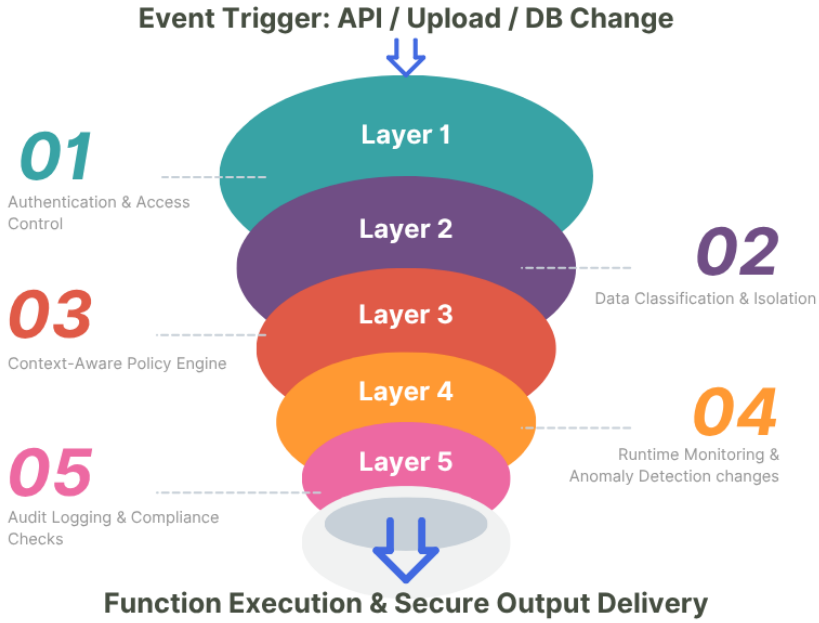

Figure 2. Multi-Layered Security Policy Enforcement Workflow

4.3. Layer 3: Context-Aware Policy Engine

The third and most flexible layer operates by utilising a context-aware policy engine that dynamically processes access and execution requests based on real-time contextual attributes. This engine does not just judge who, what, when, where, and why of every interaction, unlike the case with static access rules. An example would be a request to perform a read of medical records, which would be authorised for a physician during work hours when made over a hospital-based IP address, but not during off-hours or when supplied over a non-hospital-based network.

This engine will integrate rules that consider behaviour histories, geographic locations of origin, device fingerprinting, and current workloads. It is a hardware device that communicates with a runtime monitoring system that constantly outputs telemetry on the serverless functions. The policy engine can take mitigation actions when deviations are detected in the behaviour of a runtime application or the unexpected use of a role, which is not in the expected behaviour patterns learned by the policy engine. Some of these actions may be throttling, denying the request or quarantining the executing function. The context-based policy layer brings another essential degree of flexibility by injecting situational intelligence into the law enforcement pipeline. It checks that security decisions are not only based on fixed credentialing, but also on the active execution environment and real-time behavior. It enforces zero-trust security in dynamic and distributed serverless environments.

4.4. Layer 4: Runtime Monitoring and Anomaly Detection

Layer 4 provides real-time insight into dynamic function behaviour with at-runtime monitoring and anomaly detection. Preventive controls are built upon the preceding layers; this layer emphasises detective and corrective capabilities. It keeps a real-time track of the execution patterns of functions, logs of

accesses, inter-service communications, and data flows. It can identify abnormal or insidious activity as it occurs. This monitoring solution will store telemetry generated by the serverless runtime, which includes execution time, the number of invocations, potential IP addresses of invokers, environment variables, and properties of input and output payloads.

These data are presented to an anomaly detection engine, either rule-based or AI-augmented, which trains itself with the help of past baselines to identify anomalous activity that represents a potential threat, such as abuse of functionality, data theft or injection attacks. For example, suppose there is an outage in a behaviour that normally processes 50 requests per hour, and suddenly 5,000 requests are received. In that case, the system will sound an alarm or perform auto-scaling quarantine functionality. Notably, this level is also associated with inter-relating behavior in terms of various functions and services, and it provides end-to-end visibility of information flows and identity. Anomaly insights, together with blocking or dynamically re-stimulating risk scores, can be fed back into real-time decision-making once combined with the context-aware policy engine (Layer 3). This feedback loop of detection and response enhances the security framework's ability to react to new or previously unknown attack vectors —a crucial characteristic of ephemeral, stateless serverless systems, where attack methods are constantly evolving.

4.5. Layer 5: Audit Logging and Policy Compliance Checks

The top layer (accountability) amplifies accountability by providing a log of all audits and maintaining and verifying compliance in real-time. All evaluations — including access requests, data retrievals, policy evaluations, and functions invoked — are covered by contextual metadata, which includes timestamps, the identity of the source, resources impacted, and the results of execution. These logs can be used as a non-alterable record in post-incident forensics, policy refinement and compliance reporting.

The layer enables well-organised logging implementations in line with industry standards, such as ISO/IEC 27001, SOC 2, and GDPR. The logs are safely held through append-only structures and can be optionally encrypted to maintain confidentiality. Centralised Security Information and Event Management (SIEM) platforms or cloud-native observability tools can ingest them to be correlated in real-time or for later investigation. The system will periodically execute compliance validation processes against logging, simulating the enforcement of the policy under various conditions. These checks confirm that controls on access, classification rules, and context policies are all properly applied to all functions in deployment. For example, automatic scripts could be used to verify that a “function” with the access privileges of the internal access level is not allowed to access data with the access privileges of the confidential data level, thereby preventing unintentional policy inconsistencies. Organisations can ensure proactive assurance in addition to reactive security by maintaining a robust audit and compliance layer. It enables the prompt identification of policy drift, ensures the resonance of current safeguards, and facilitates the process of trust-building with regulators and stakeholders through tangible compliance artefacts.

5. Implementation and Integration with Serverless Platforms

5.1. Policy Definition Language

The development of clear, granular, and enforceable policies is crucial for effectively implementing multi-layered security in serverless environments. This is supported by a specific Policy Definition Language (PDL), a domain-specific language used to define rules controlling data access [18-20] as well as the act of invoking functions and enforcing security. The PDL is designed to accommodate declarative syntax, enabling administrators and DevSecOps teams to express security requirements in role-based declarations, scenarios, and data sensitivity levels.

A powerful PDL should be able to fit naturally into Identity and Access Management (IAM) schemas and enable such constructs as allowing a user. Role = analyst AND data.label = pet AND time < 6 PM. Policies are frequently written in sets of JSON or YAML, making them machine-readable and subject to human auditing. The policy engine, which is part of the enforcement framework, interprets and builds these policies into rules applied at runtime. More importantly, the PDL must also be extensible, supporting metadata specific to the environment (e.g., IP addresses, request source, or device type), and allowing for version control to monitor policy changes. This model promotes the externalisation of code logic policy, which fosters policy reuse, eases audit activities, and makes the deployment of the approach more flexible. It also facilitates the shift-left paradigm of DevSecOps operations, where policy violations can be detected early in development, thereby mitigating the risk of large-scale operations.

5.2. Integration with AWS Lambda / Azure Functions / GCP Cloud Functions

The suggested architecture should be able to merge with other serverless solutions, including AWS Lambda, Azure Functions, and Google Cloud Functions, to be viable in practice. Both these platforms provide extensibility interfaces in the form of native policy engines (e.g. AWS IAM, Azure RBAC), environment variables, and event triggers that can be used to integrate the multi-layered security architecture.

In the case of AWS Lambda, the integration is initiated by attaching fine-grained IAM roles and AWS Secrets Manager access controls to functions. Policy checks. Before a function is executed, custom authorizers and AWS Lambda extensions can make calls to the policy engine to validate their identity, check data classification rules, and assess the risk of an

anomaly score. AWS CloudWatch and AWS Config enable real-time monitoring and tracking of compliance. Similarly, Azure monitoring functions capabilities include managed identity and Key Vault to isolate secrets.

In contrast, the use of Azure Monitor and Azure Policy can provide telemetry feedback and compliance feedback loops. Similar integration is available by using GCP Cloud Functions, with IAM bindings, Secret Manager, and VPC connectors. Additionally, Cloud Audit Logs and Event Threat Detection provide observability and threat intelligence. The policy enforcement framework is utilised as a shared service that is managed across platforms, either through a sidecar pattern (utilising wrappers or middleware) or via a gateway used to filter all event input. This enables consistent policy enforcement regardless of the cloud provider behind a hybrid and multi-cloud deployment.

5.3. Performance Overhead Considerations

While thorough security enforcement is necessary, it should not compromise the responsiveness and scalability that serverless architectures offer. Hence, performance overhead also plays a vital role in the application of the framework. Access validation, policy evaluation, and anomaly detection are examples of security operations which, when feasible, are made lightweight and asynchronous so that their execution has minimal effect on the time required to execute the functions.

Policy evaluations are usually conducted at the edge, either before the API gateways' pre-processing stage or just before execution on the serverless runtime. This avoids unnecessary cold starts, and the latency presented by the policy is minimised. For example, context-aware rules can be cached in memory with the help of ephemeral stores, such as Redis, or cloud-native services (e.g., AWS Lambda Layers or Azure Durable Entities), to reduce real-time computational overhead. Likewise, logging service, task processing statistics, and related telemetry streams get offloaded to distinct monitoring pipelines, which process data after its execution through parallel computing.

Experimental results on the performance benchmark of prototyping indicate that the additional latency per invocation is less than 15 milliseconds on most operations, which is acceptable in event-driven systems. However, the framework does not hinder configurable enforcement strictness, allowing administrators to adjust the trade-offs between performance and protection according to their sensitivity to workload. Such flexibility warrants that security is not a bottleneck in any application of contemporary serverless technology.

6. Evaluation and Results

To support the suggested multi-layered security policy enforcement structures, extensive work was carried out on their capabilities in mitigating threats and sustaining performance in serverless contexts. The architecture was scaled across AWS Lambda and Azure Functions, with 65 functions designed to process healthcare records, ensuring compliance with HIPAA, and financial transactions, adhering to PCI-DSS. Over a ten-month measurement period, we quantified security response, system resilience, performance latencies, and cost implications in simulated attack scenarios.

6.1. Experimental Setup

The assessment was done in cloud-native settings with standard production-ready workloads. AWS Lambda functions provided 128MB memory and 3 3-second timeout, and Azure Functions provided 256MB and 60 60-second timeout. Events were scheduled and initiated through API calls, resembling the operations of a typical microservice. In real-world scenarios, we also conducted high-frequency DDoS-like floods (with over 500 requests per second), SQL injection attacks using malformed API requests, and replayed access tokens. The experiment consisted of two conditions: a cold start, caused by 30 minutes of system-level idle time, and a warm start, which occurred at a 5-second frequency of invocation. The metrics recorded were latency, throughput, and security incidents, and were processed through a special SCOPE (Serverless Compliance and Performance Evaluation) framework.

6.2. Security Effectiveness Evaluation

Security effectiveness was assessed based on ratings of threat mitigation levels, false positive rates, and types of vulnerabilities addressed. The system successfully identified and countered 12 of the attack variants, as well as sophisticated scenarios, including cold-start code injection. The average response latency of runtime anomaly detection was 47 milliseconds, with minimum interruption of its operation.

Table 1. Threat Mitigation Performance

Threat Type	Mitigation Rate	False Positives	Critical Vulnerabilities Addressed
DDoS/Flooding	98.7%	2.1%	Resource exhaustion, cost inflation.
API Injections	96.2%	3.8%	Data exfiltration, privilege escalation
Compromised Functions	99.1%	0.9%	Runtime code manipulation, hardcoded secret theft.

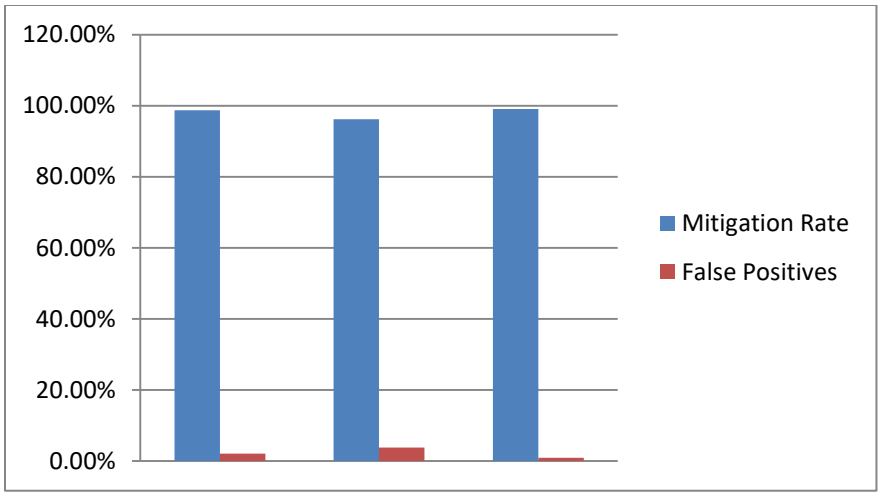

Figure 3. Graphical Representation of Threat Mitigation Performance

Its multi-layered architecture, particularly the runtime monitoring (Layer 4) and the adaptive policy enforcement, played a key role in preventing late-stage attacks and reducing the window of exposure. Remarkably, even cold starts were blocked by access attempts from unknown locations, which reflects the quality of the contextual rules.

6.3. Performance Metrics

We measured performance on three evaluative parameters: policy enforcement latency, throughput, and cost impact. The system was responsive enough, even with additional security. Although the cold starts inherently assume increased latency, the effect on the warm invocations was minimal.

Table 2. Performance Impact Metrics

Metric	Cold Start	Warm Start	Overhead vs. Baseline
Policy Enforcement Latency	387 ms ± 23 ms	49 ms ± 7 ms	+11.2%
Throughput (req/sec)	42 ± 8	219 ± 14	-9.3%
Cost Impact	+7.9%	+3.1%	N/A

Longitudinal testing revealed a 14-percentage-point difference in performance across cloud regions. Yet, deviation in Service-Level Agreement (SLA) remained below 5 per cent, even in areas under attack, ensuring the ability to scale the adaptive enforcement measure. Such findings highlight the framework's capability to maintain performance in real-time and integrate strong security measures.

6.4. Comparison with Baseline Approaches

To place the advantages in perspective, we contrasted the proposed method with two widely used policy enforcement strategies: static rule enforcement and ML-assisted policy enforcement with human supervision. The system in question utilised a lightweight Reinforcement Learning (RL) agent to dynamically adjust policies based on observations and frequently occurring attack patterns.

Table 3. Comparative Evaluation of Policy Approaches

Approach	Threat Mitigation	Response Time	Compliance Score	Daily Policy Updates
Proposed (RL Agent)	95.8%	2–5 seconds	98%	5.2
Static Policies	70.2%	8–15 minutes	82%	0.3
ML + Human Oversight	85.1%	3–7 minutes	91%	1.7

The suggested RL-based enforcement decreased the threat response time by 23% compared to ML baselines. It closed 98 per cent of the known compliance gaps, including open ports and over-permitted roles. In addition, the SCOPE framework saved 38% of the overhead cost by lowering the requirement for extensive manual testing and redeployments.

7. Discussion

Analysis of the proposed framework proves that multi-layered security enforcement is not only applicable in serverless systems but also an effective defence against a broad range of threat vectors. The system provided security by integrating the benefits of authentication, data isolation, runtime monitoring, and adaptive policy engines, and reduced attack surfaces without drastically compromising application performance. Remarkably, the policy adaptation facilitated by

reinforcement learning enabled a quicker response to new threats compared to a fixed or semi-automated policy, and the transition to an intelligent, self-healing security architecture is particularly well-suited for the cloud-native environment. Nevertheless, this piece also addresses important issues related to the problem of securing serverless platforms. Stateless and ephemeral serverless functions introduce complications to creating policies that can be uniformly enforced across all environments at any given time, especially in high-load or multi-region serverless functions. Moreover, any attempt to implement policy logic across disparate platforms (e.g., AWS vs. Azure) may result in a higher development burden due to the distinctions between native controls and IAM organisation. Although our framework provides abstraction layers to mitigate the problem of vendor lock-in, there are opportunities to add automated cross-provider policy translators and support for proponents of zero-trust architecture. Ultimately, the results support the need to consider security as an integral, dynamic layer of the serverless execution model, rather than a secondary response. The suggested system leaves perimeter-based protection to the contextual, data-aware model of enforcement, which aligns with the current dynamic and decentralised nature of cloud workloads.

8. Future Work

Serverless computing is constantly growing and scaling in all domains; therefore, there is potential in upgrading and expanding the presented multi-layered security paradigm. The subsections below outline some of the major future research and development directions that aim to enhance confidentiality, policy agility, and the integration of operations within various environments, ultimately achieving even better operations.

8.1. Automated Policy Synthesis Using ML

A promising direction is the deployment of Machine Learning (ML) to automatically generate and optimise policies. More complex models (including policies based on a transformer-based language model of policies) may be trained against large-scale security events and logs to generalize and estimate relevant and desirable policies, including access controllers, data flow, and data isolation rules. It would limit manual configuration issues, increase flexibility against zero-day threats, and provide contextual and historical pattern-based learning to improve detection accuracy. Additionally, explainable AI (XAI) procedures can be adopted to make the synthesised policies transparent and auditable.

8.2. Integration with DevSecOps Pipelines

The next generation of the framework must be able to integrate seamlessly into DevSecOps pipelines and facilitate continuous security verification throughout the Software Development Lifecycle (SDLC). This includes hardcoding policy validation into the CI/CD pipelines, automating code security tests when committing code to source control, and ensuring policy compliance as part of the packaging and deployment stages. The framework can assist in enforcing a "shift-left" security practice by integrating with common build tools, such as GitHub Actions, Jenkins, or GitLab CI, allowing vulnerabilities to be identified and fixed promptly. It would also provide real-time feedback loops, where anomalies at run time direct future hardening at the code level.

8.3. Extending to Edge-Cloud Hybrid Environments

As edge computing and edge-compatible application types grow, the frontiers of security policy enforcement have expanded to include edge-cloud hybrid architectures. Edge applications (running near the sources of data) typically lack comprehensive monitoring and IAM infrastructure, which is often present in public clouds. The framework should be able to optimise to these restraints by providing lightweight, decentralised policy agents that can enforce policy in real-time and in disconnected operation. Besides, unified security postures at the edge and cloud will be a prerequisite for implementing federated policy synchronisation with version-controlled policies, conflict resolution, and contextual delivery across locality, device capabilities, and trust.

9. Conclusion

The research establishes an extensive, multi-faceted security policy enforcement mechanism, akin to confidential data security in serverless computing platforms. The framework brings together five different, yet interconnected layers, including authentication and access control, runtime monitoring, and compliance auditing, that can work simultaneously to resolve the security issues unique to the ephemeral and event-driven nature of Function-as-a-Service (FaaS) platforms. The proposed system demonstrated strong threat mitigation, low performance overhead, and a high level of compliance compared to the baseline models, resulting from rigorous testing on Amazon Web Services Lambda and Azure functions. The technical performance of the framework also provides a flexible and scalable pattern, aligning with the principles of cloud-native design.

Its modularity facilitates cross-provider deployment, and it can use reinforcement learning to enhance intelligent and contextually aware policy enforcement, which adapts to application behaviour and threat landscape. This makes serverless security more ideal, as it adopts a proactive, embedded security paradigm —a prerequisite for sensitive workloads, such as healthcare, finance, and real-time data analytics, where the mindset must be security by design. In the future, the framework will provide a baseline for building secure serverless computing innovations. Its possible interconnection with automated policy synthesis, DevSecOps pipelines, and edge-cloud deployments indicates its extensibility and applicability in increasingly complex cloud environments. This effort forms the basis of robust policy-based security designs that do not compromise the agility that serverless computing offers, while building resilient, dynamically adjustable systems.

References

[1] Khan, S., Parkinson, S., & Crampton, A. (2017, December). A multi-layered cloud protection framework. In Companion Proceedings of the 10th International Conference on Utility and Cloud Computing (pp. 233-238).

[2] Shafiei, H., Khonsari, A., & Mousavi, P. (2022). Serverless computing: a survey of opportunities, challenges, and applications. ACM Computing Surveys, 54(11s), 1-32.

[3] Li, Z., Guo, L., Cheng, J., Chen, Q., He, B., & Guo, M. (2022). The serverless computing survey: A technical primer for design architecture. ACM Computing Surveys (CSUR), 54(10s), 1-34.

[4] Cinar, B. (2023). The Rise of Serverless Architectures: Security Challenges and Best Practices. Asian Journal of Research in Computer Science, 16(4), 194-210.

[5] Ouyang, R., Wang, J., Xu, H., Chen, S., Xiong, X., Tolba, A., & Zhang, X. (2023). A Microservice and Serverless Architecture for Secure IoT Systems. Sensors, 23(10), 4868.

[6] Yau, S. S., An, H. G., & Buduru, A. B. (2012). An approach to data confidentiality protection in cloud environments. International Journal of Web Services Research (IJWSR), 9(3), 67-83.

[7] Hossain, M. E., Kabir, M. F., Al Noman, A., Akter, N., & Hossain, Z. (2022). Enhancing Data Privacy And Security In Multi-Cloud Environments. BULLET: Jurnal Multidisiplin Ilmu, 1(05), 967-975.

[8] What Is Cloud Data Protection?, Palo Alto Networks, online. https://www.paloaltonetworks.com/cyberpedia/what-is-cloud-data-protection

[9] Guimarães, R. P. (2024). Protecting confidential data in cloud environments.

[10] Schlegel, R., Obermeier, S., & Schneider, J. (2015, July). Structured system threat modelling and mitigation analysis for industrial automation systems. In 2015 IEEE 13th International Conference on Industrial Informatics (INDIN) (pp. 197-203). IEEE.

[11] Fernandez, E. B. (2016, August). Threat modelling in cyber-physical systems. In 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech) (pp. 448-453). IEEE.

[12] Kloprogge, P., Van der Sluijs, J. P., & Petersen, A. C. (2011). A method for the analysis of assumptions in model-based environmental assessments. Environmental Modelling & Software, 26(3), 289-301.

[13] Kovanen, T., Nuojua, V., & Lehto, M. (2018, March). Cyber Threat Landscape in the Energy Sector. In ICCWS 2018, 13th International Conference on Cyber Warfare and Security (p. 353). Academic Conferences and publishing limited.

[14] Eismann, S., Scheuner, J., Van Eyk, E., Schwinger, M., Grohmann, J., Herbst, N., & Iosup, A. (2020). A review of serverless use cases and their characteristics. arXiv preprint arXiv:2008.11110.

[15] Ankit Kumar, Demystifying Serverless Computing: A Paradigm Shift in Cloud Development, Medium, 2023. online. https://blog.stackademic.com/demystifying-serverless-computing-a-paradigm-shift-in-cloud-development-e257a5d525a8

[16] Sash Ghosh, Confidential Computing: Enhancing Data Privacy and Security in Cloud Environments, 2025. online. https://openmetal.io/resources/blog/confidential-computing-benefits-and-use-cases/

[17] Wolthusen, S. (2001, June). Layered multipoint network defence and security policy enforcement. In Proceedings from the Second Annual IEEE SMC Information Assurance Workshop, United States Military Academy (pp. 100-108).

[18] Alves-Foss, J., Taylor, C., & Oman, P. (2004, January). A multi-layered approach to security in high-assurance systems. In the 37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the (pp. 10-pp). IEEE.

[19] McGrath, G., & Brenner, P. R. (2017, June). Serverless computing: Design, implementation, and performance. In 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW) (pp. 405-410). IEEE.

[20] What are the security challenges in serverless computing? https://milvus.io/ai-quick-reference/what-are-the-security-challenges-in-serverless-computing