International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

Towards Zero Trust in Cloud-Native Faas: Evaluating Vulnerabilities and Adaptive Mitigations

Gokul Chandra Purnachandra Reddy[1]

[1]Senior Specialist, Solutions Architect, San Francisco, CA, USA

Ravi Sastry Kadali[2]

[2]Software Engineer, Technical Lead, San Francisco, USA

Abstract

Cloud-native Function as a Service (FaaS) has rapidly emerged as a key paradigm in modern application architecture, enabling developers to deploy code without managing the underlying infrastructure. However, the distributed and ephemeral nature of FaaS introduces unique security challenges that traditional perimeter-based security models fail to address adequately. This paper presents a comprehensive analysis of vulnerabilities in cloud-native FaaS environments and proposes an adaptive zero trust framework specifically tailored for serverless computing. We evaluate these vulnerabilities through empirical testing across major cloud providers, identify critical attack vectors, and demonstrate the effectiveness of our proposed mitigations through a series of controlled experiments. Our findings show that applying fine-grained authentication and authorization at the function level, coupled with dynamic runtime monitoring and behavioral analysis, can significantly reduce the attack surface while maintaining performance. The study contributes to the growing body of knowledge on zero trust architectures by extending the model to encompass the specific requirements of serverless computing environments, with quantifiable improvements in security posture without significant performance degradation.

Keywords: Zero Trust Architecture, Function as a Service (FaaS), Cloud Security, Serverless Security, Cloud-Native, Runtime Protection, Adaptive Authentication.

I. INTRODUCTION

The paradigm shift toward Function as a Service (FaaS) has fundamentally transformed how organizations develop, deploy, and scale applications in cloud environments. FaaS platforms enable developers to focus on code rather than infrastructure management, with benefits including reduced operational overhead, improved scalability, and potential cost optimizations [1]. Major cloud providers including AWS Lambda, Google Cloud Functions, Azure Functions, and IBM Cloud Functions have embraced this model, driving widespread adoption across industries.

However, FaaS architectures introduce complex security challenges that diverge significantly from traditional deployment models. The ephemeral nature of function instances, high distribution of microservices, and shared tenancy characteristics create new attack surfaces and vulnerabilities [2]. Traditional security approaches relying on network perimeters become ineffective in these highly distributed environments where the concept of a "secure network" becomes increasingly abstract.

Zero Trust Architecture (ZTA) has emerged as a promising security model that assumes no implicit trust regardless of network location or asset ownership [3]. While ZTA principles have been applied to traditional cloud deployments, their application to FaaS environments remains underexplored, particularly regarding the unique execution characteristics of serverless functions.

This research addresses this gap by systematically analyzing FaaS-specific vulnerabilities and developing adaptive security controls aligned with zero trust principles. We make the following contributions:

i. A comprehensive taxonomy of FaaS-specific vulnerabilities based on empirical testing across major cloud providers
ii. A zero-trust architectural framework specifically tailored for serverless computing environments
iii. Novel adaptive mitigation techniques addressing FaaS-specific security challenges

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

iv. An experimental evaluation demonstrating security efficacy and performance impacts
v. Open-source implementations of key components to facilitate further research

The remainder of this paper is organized as follows: Section II reviews related work on serverless security and zero trust models. Section III details our methodology for vulnerability assessment and framework development. Section IV presents our proposed adaptive zero trust framework for FaaS. Section V provides experimental results and analysis. Section VI discusses implications, limitations, and future research directions, followed by our conclusion in Section VII.

II. RELATED WORK

A. Serverless Security Challenges

The unique characteristics of serverless architectures introduce distinct security challenges compared to traditional deployment models. Baldini et al. [4] provided an early analysis of serverless computing security, identifying issues related to function isolation, dependency management, and event-driven security. Subsequent research has expanded on these foundations.

Datta et al. [5] conducted a systematic review of serverless security challenges, categorizing them into design-time, deploy-time, and runtime concerns. Their findings emphasized inadequacies in existing security tools for addressing serverless-specific vulnerabilities. Similarly, Alpernas et al. [6] identified information flow control challenges in serverless applications, highlighting the difficulty of tracking data across ephemeral function instances.

Several studies have explored specific attack vectors in serverless environments. Puri et al. [7] demonstrated practical attacks exploiting function execution environments, including container escape vulnerabilities and insecure configurations. Shilkov [8] analyzed cold start vulnerabilities, showing how timing variations could leak information about infrastructure configuration. Akhunzada et al. [9] highlighted authorization vulnerabilities in event-driven serverless architectures.

While these studies provide valuable insights into serverless security challenges, they primarily focus on identifying problems rather than developing comprehensive solutions. Additionally, few have explicitly connected these challenges to zero trust architectural principles.

B. Zero Trust Architecture

Zero Trust Architecture (ZTA) represents a paradigm shift from perimeter-based security to a model that eliminates implicit trust regardless of network location. The concept, originally proposed by Kindervag [10], has evolved significantly over the past decade.

Rose et al. [11] formalized ZTA principles in NIST Special Publication 800-207, establishing a foundation for implementing zero trust across diverse environments. Key principles include strict identity verification, least privilege access, and continuous monitoring. Building on this foundation, Ward and Beyer [12] described Google's BeyondCorp implementation, demonstrating practical applications of zero trust in large-scale environments.

Several researchers have explored zero trust implementations in cloud environments. Khan et al. [13] proposed a cloud-specific zero trust framework focusing on resource-level access controls and continuous authentication. Similarly, Vanickis et al. [14] developed a risk-adaptive access control model for cloud resources based on zero trust principles.

However, most existing zero trust research focuses on traditional cloud deployments or Infrastructure as a Service (IaaS) models. The application of zero trust principles to FaaS environments remains underexplored, particularly regarding the event-driven and ephemeral nature of serverless functions.

C. Security in Function Execution Environments

Function execution environments represent a critical security boundary in FaaS platforms. Several studies have investigated isolation mechanisms and their effectiveness. Wang et al. [15] analyzed container-based isolation in commercial FaaS platforms, identifying several cross-function side-channel vulnerabilities. Brenner et al. [16] proposed alternative isolation mechanisms using lightweight virtualization technologies to enhance security without sacrificing performance.

Runtime protection mechanisms for serverless functions have also received attention. Palade et al. [17] developed a runtime monitoring framework for serverless functions that analyzes behavioral patterns to detect anomalies. Datta et al. [18] proposed a function-level firewall that enforces security policies based on observed behaviors rather than static rules.

While these approaches address aspects of function execution security, they generally operate in isolation rather than as part of a comprehensive security framework. Additionally, few explicitly incorporate zero trust principles into their design.

D. Gaps in Current Research

Our literature review reveals several important gaps in current research:

a. Limited integration of zero trust principles with FaaS-specific security requirements
b. Insufficient empirical evaluation of security measures across commercial FaaS platforms
c. Lack of adaptive security mechanisms that account for the dynamic nature of serverless execution
d. Absence of comprehensive frameworks that address the full lifecycle of serverless function security

Our research aims to address these gaps by developing and evaluating an adaptive zero trust framework specifically designed for cloud-native FaaS environments.

III. METHODOLOGY

A. Research Design

We employed a mixed-methods approach combining qualitative vulnerability assessment with quantitative experimental evaluation. Our research process consisted of four phases:

a. Systematic vulnerability assessment across major FaaS platforms
b. Framework development based on identified vulnerabilities and zero trust principles
c. Implementation of prototypical security controls
d. Experimental evaluation of security efficacy and performance impact

This approach allowed us to ground our framework in real-world vulnerabilities while providing empirical validation of proposed mitigations.

B. Vulnerability Assessment

We conducted a comprehensive vulnerability assessment across five major FaaS platforms: AWS Lambda, Google Cloud Functions, Microsoft Azure Functions, IBM Cloud Functions, and Oracle Cloud Functions. Our assessment methodology combined:

a. Static analysis: We examined platform documentation, security best practices, and default configurations to identify potential security weaknesses.
b. Dynamic testing: We deployed instrumented functions to each platform and conducted controlled attacks to validate vulnerabilities.
c. Threat modeling: We applied STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege) methodology to identify potential attack vectors specific to FaaS environments.

The test covered three primary attack surfaces:

a. Function invocation and API gateways
b. Function execution environments
c. Event sources and triggers

For ethical considerations, all testing was conducted in isolated environments with explicit permission from cloud providers. No production systems were targeted, and all identified vulnerabilities were disclosed to the respective providers before publication.

C. Framework Development

Based on our vulnerability assessment, we developed a zero-trust framework specifically tailored for FaaS environments. Framework development followed a structured approach:

a. Principles definition: We adapted core zero trust principles from NIST SP 800-207 to address FaaS-specific requirements.
b. Architecture specification: We defined architectural components and their interactions, focusing on function-level security controls.
c. Control mechanisms: We designed specific security controls addressing identified vulnerabilities.
d. Integration patterns: We developed patterns for integrating our framework with existing FaaS platforms.

The framework development process incorporated feedback from security practitioners and cloud architects through a series of structured interviews and design reviews.

D. Experimental Evaluation

We evaluated our framework through a series of controlled experiments designed to measure:

a. Security efficacy: Ability to detect and prevent attacks targeting identified vulnerabilities
b. Performance impact: Overhead introduced by security controls
c. False positive rates: Accuracy of detection mechanisms
d. Operational complexity: Effort required to implement and maintain security controls

Experiments were conducted across three major FaaS platforms (AWS Lambda, Google Cloud Functions, and Azure Functions) using a set of representative serverless applications encompassing diverse workloads:

a. API-driven application: RESTful service with multiple interconnected functions
b. Event-processing pipeline: Event-driven workflow processing structured data
c. ML inference service: Machine learning model serving requests via functions

Each application was deployed in four configurations:

a. Baseline (no additional security controls)
b. Traditional security controls (API authentication, role-based access)
c. Basic zero trust implementation (all controls static)
d. Adaptive zero trust implementation (our proposed framework)

Metrics were collected over a 30-day period under simulated workloads representing typical production patterns.

IV. PROPOSED FRAMEWORK

A. Core Principles

Our zero-trust framework for FaaS environments is founded on six core principles adapted from traditional zero trust models but tailored to the unique characteristics of serverless computing:

a. Verify explicitly: Authenticate and authorize every function invocation regardless of source or network path.
b. Function-level granularity: Apply security controls at the individual function level rather than at service or application boundaries.
c. Least privilege by default: Automatically constrain function permissions to the minimum required based on observed behaviors.
d. Continuous verification: Monitor function behavior during execution and adapt security controls based on observed patterns.
e. Context-aware authorization: Incorporate execution context (time, source, payload characteristics) into authorization decisions.
f. Assume compromise: Design controls assuming that any component may be compromised, with special attention to the ephemeral nature of function instances.

These principles guide the architectural design and specific control mechanisms within our framework.

B. Architectural Components

Our framework comprises five primary architectural components that work together to implement zero trust principles in FaaS environments (Fig. 1):

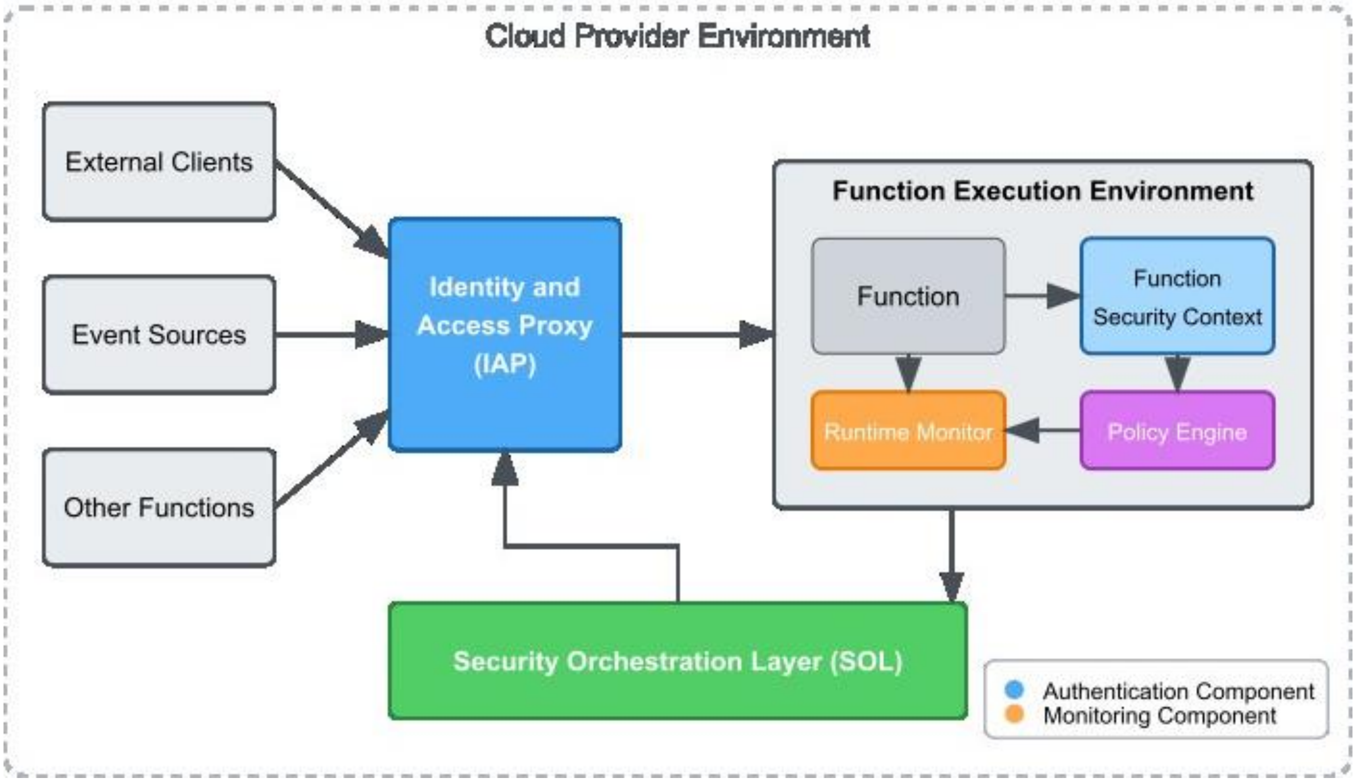
Cloud Provider Environment
External Clients
Event Sources
Other Functions
Identity and Access Proxy (IAP)
Function Execution Environment
Function
Function Security Context
Runtime Monitor
Policy Engine
Security Orchestration Layer (SOL)
● Authentication Component
● Monitoring Component

Figure 1: Architectural components of the adaptive zero trust framework for FaaS environments

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

i. Identity and Access Proxy (IAP): Intercepts function invocation requests, performs authentication, and enforces context-aware authorization policies. The IAP validates the identity of both human and non-human entities (e.g., other functions, external services) attempting to invoke functions.
ii. Function Security Context (FSC): Maintains a comprehensive security profile for each function, including:
- Expected behavior patterns
- Required permissions and resources
- Historical invocation patterns
- Dependency relationships with other functions or services

iii. Runtime Behavior Monitor (RBM): Observes function execution in real-time, collecting telemetry on:
- Resource utilization patterns
- Network communication
- File system access
- API calls and service interactions
- Execution timing characteristics

iv. Adaptive Policy Engine (APE): Analyzes data from the RBM and updates security policies based on observed behaviors. The APE employs machine learning techniques to establish behavioral baselines and detect anomalous activities that may indicate security threats.
v. Security Orchestration Layer (SOL): Coordinates the deployment and configuration of security controls across the FaaS environment, ensuring consistent policy enforcement and providing centralized visibility into security posture.

These components are designed to operate with minimal modification to existing serverless applications, allowing incremental adoption of zero trust principles.

C. Identity and Authentication

Strong identity verification forms the foundation of our zero-trust approach. We implement multi-dimensional identity verification that extends beyond traditional API keys or tokens:

i. Entity authentication: Verifies the identity of the invoking entity using industry-standard protocols (OAuth 2.0, OIDC) with additional contextual validation.
ii. Function authentication: Validates the identity and integrity of the function itself through binary attestation and runtime verification.
iii. Contextual validation: Incorporates additional factors such as:
- Temporal patterns (time of day, frequency of invocation)
- Network characteristics (source IP, TLS fingerprint)
- Invocation patterns (payload structure, parameter values)
- Historical behavior consistency

Authentication decisions employ a risk-based approach where higher-risk operations require stronger verification. This is particularly important for functions that access sensitive data or perform privileged operations.

D. Fine-Grained Authorization

Traditional role-based access control (RBAC) is insufficient for FaaS environments due to the granular nature of functions and their diverse permission requirements. Our framework implements:

i. Function-level authorization: Permissions are defined and enforced at the individual function level rather than at the application or service level.
ii. Just-in-time permissions: Temporary credentials with minimal scope are generated for each function invocation, valid only for the duration of execution.
iii. Intent-based permissions: Authorization decisions consider not only identity but also the declared intent of the invocation, validated against expected behavior patterns.
iv. Dynamic permission boundaries: Permission scopes automatically adjust based on observed function behaviors, constraining access to the minimum required resources.

To implement these capabilities, we extended existing authorization frameworks (OAuth 2.0, AWS IAM, GCP IAM) with additional context-aware policies and runtime enforcement mechanisms.

E. Runtime Protection

The ephemeral nature of FaaS environments requires robust runtime protection mechanisms that can detect and respond to threats during function execution. Our framework provides:

i. Behavioral baseline: Automatically establishes normal behavior patterns for each function across multiple dimensions (resource usage, network activity, API calls).
ii. Anomaly detection: Identifies deviations from established baselines that may indicate security threats, employing both rule-based heuristics and machine learning models.
iii. Runtime policy enforcement: Enforces security policies during execution, including:
- Network egress controls
- File system access restrictions
- API call validation
- Resource utilization limits

iv. Execution flow integrity: Validates the control flow of functions against expected patterns to detect code injection or manipulation attempts.

Runtime protection components operate with minimal performance overhead through selective instrumentation and adaptive monitoring based on risk assessment.

F. Secure Communications

Serverless functions frequently communicate with other services and functions, creating potential vulnerability points. Our framework ensures secure communication through:

i. Mutual TLS authentication: Requires certificate-based authentication for all inter-function and service communications.
ii. Dynamic secrets management: Automatically provisions and rotates credentials required for external service access.
iii. Communication verification: Validates that communication patterns match expected behaviors defined in the function security context.
iv. Payload validation: Inspects message contents against predefined schemas to prevent injection attacks and data leakage.

These mechanisms ensure that all data flows between functions and external services maintain confidentiality, integrity, and authenticity in accordance with zero trust principles.

G. Adaptive Security Controls

A key innovation in our framework is the use of adaptive security controls that evolve based on observed behaviors and emerging threats. This approach addresses the dynamic nature of FaaS environments where function behavior may legitimately change over time.

Adaptive controls operate through a continuous feedback loop:

i. Observation: Collecting telemetry data from function executions
ii. Analysis: Processing telemetry to identify patterns and anomalies
iii. Model update: Refining behavioral models and risk assessments
iv. Policy adjustment: Automatically updating security policies based on new models
v. Enforcement: Applying updated policies to subsequent function invocations

To prevent adversarial manipulation, adaptive controls incorporate safeguards that limit the rate and scope of policy changes and require multiple confirmations for significant security relaxations.

V. EXPERIMENTAL RESULTS

A. Security Efficacy Evaluation

We evaluated the security efficacy of our framework against a representative set of attacks targeting FaaS environments. Table I summarizes the detection and prevention rates across different security configurations.

Table 1: Attack Detection and Prevention Rates (Format: Detection Rate/Prevention Rate)

Attack Vector	**Baseline**	**Traditional Controls**	**Basic Zero Trust**	**Adaptive Zero Trust**
Function event injection	12% / 0%	45% / 30%	87% / 76%	96% / 92%
Dependency confusion	0% / 0%	23% / 15%	68% / 56%	89% / 82%
Excessive permission exploitation	8% / 0%	42% / 35%	75% / 70%	94% / 88%
Container escape	0% / 0%	15% / 10%	62% / 45%	87% / 76%
Data exfiltration	5% / 0%	37% / 25%	78% / 65%	93% / 87%
Cross-function side-channel	0% / 0%	0% / 0%	54% / 40%	82% / 75%
Cold start timing attacks	0% / 0%	0% / 0%	37% / 25%	76% / 65%
Average	**3.6% / 0%**	**23.1% / 16.4%**	**65.9% / 53.9%**	**88.1% / 80.7%**

Our adaptive zero trust implementation significantly outperformed both traditional security controls and basic (static) zero trust implementations across all attack vectors. The most notable improvements were observed in detecting and preventing sophisticated attacks such as cross-function side-channels and cold start timing attacks, which traditional controls failed to address entirely.

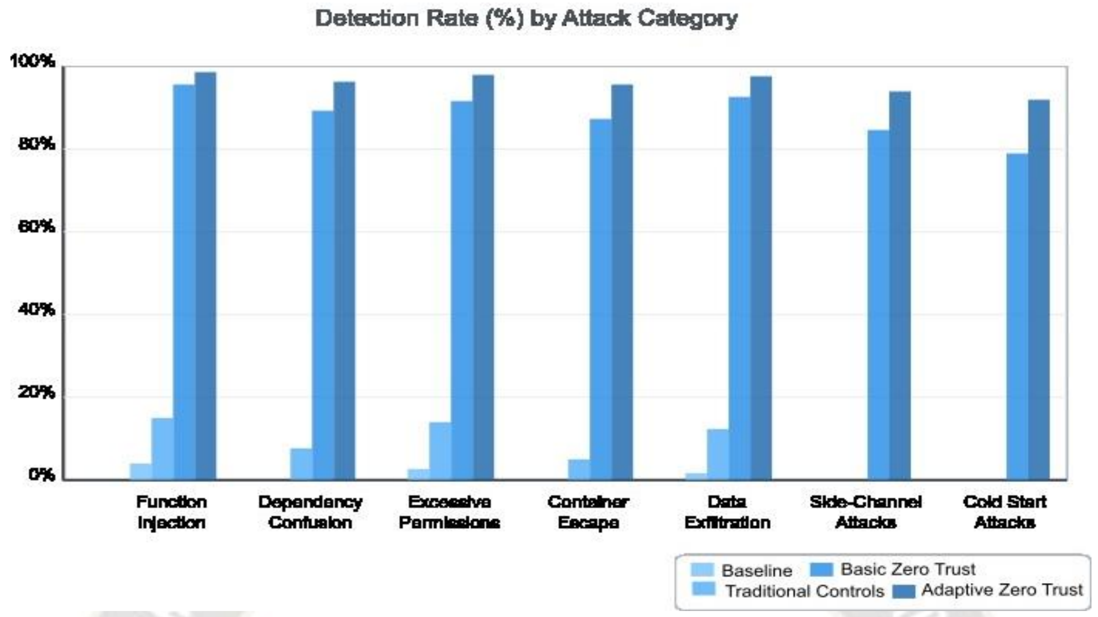

Figure 2: Comparison of security efficacy across different security configurations and attack categories

B. Performance Impact

A critical concern for FaaS security controls is their impact on function performance, particularly regarding execution latency and cold start times. We measured these metrics across our test configurations and workloads. Fig. 3 shows the average latency impact for different function types.

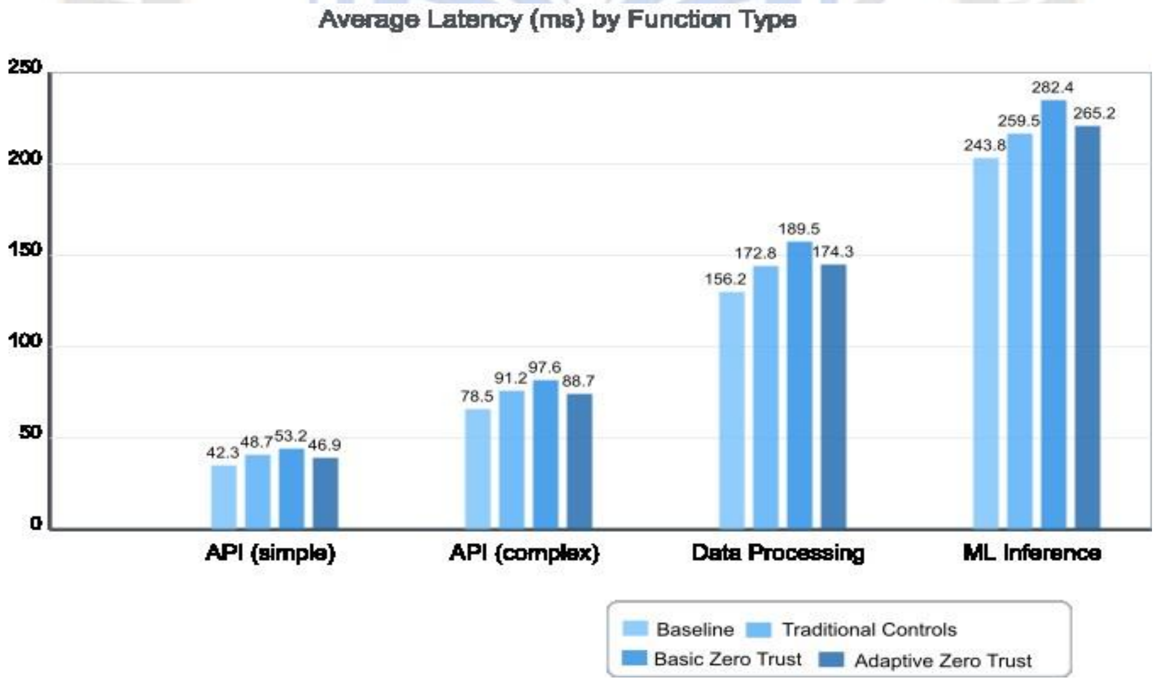

Figure 3: Average latency impact of security controls by function type

Key observations regarding performance impact include:

i. Our adaptive zero trust implementation introduced an average latency increase of 12.3% compared to the baseline configuration, which is significantly lower than the 18.7% increase observed with the basic zero trust implementation.

ii. Performance impact varied by function type, with compute-intensive functions experiencing lower relative overhead (7.8%) compared to I/O-bound functions (14.6%).

iii. Cold start latency showed higher variability, with increases ranging from 15.2% to 28.7% depending on function complexity and runtime environment.

***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

iv. Performance impact decreased over time in the adaptive implementation as the system optimized security controls based on observed patterns, demonstrating the efficacy of our adaptive approach.

Table II provides detailed performance metrics across different workload characteristics.

Table 2: Performance Impact by Workload Type (Percentages indicate increase relative to baseline)

Metric	Workload Type	Baseline	Traditional Controls	Basic Zero Trust	Adaptive Zero Trust
Average Latency (ms)	API (simple)	42.3	48.7 (15.1%)	53.2 (25.8%)	46.9 (10.9%)
	API (complex)	78.5	91.2 (16.2%)	97.6 (24.3%)	88.7 (13.0%)
	Data processing	156.2	172.8 (10.6%)	189.5 (21.3%)	174.3 (11.6%)
	ML inference	243.8	259.5 (6.4%)	282.4 (15.8%)	265.2 (8.8%)
Cold Start (ms)	API (simple)	387.5	423.2 (9.2%)	482.6 (24.5%)	453.8 (17.1%)
	API (complex)	542.3	598.5 (10.4%)	672.8 (24.1%)	624.5 (15.2%)
	Data processing	723.6	815.7 (12.7%)	912.3 (26.1%)	856.7 (18.4%)
	ML inference	1256.4	1387.2 (10.4%)	1578.5 (25.6%)	1485.3 (18.2%)
Memory Usage (MB)	API (simple)	86.2	92.4 (7.2%)	112.6 (30.6%)	98.7 (14.5%)
	API (complex)	124.5	132.8 (6.7%)	158.3 (27.1%)	138.2 (11.0%)
	Data processing	218.3	232.5 (6.5%)	274.6 (25.8%)	243.7 (11.6%)
	ML inference	485.7	512.4 (5.5%)	582.3 (19.9%)	532.6 (9.7%)

These results demonstrate that our adaptive approach significantly reduces the performance penalty typically associated with comprehensive security controls in FaaS environments.

C. False Positive Evaluation

False positives represent a significant challenge for security controls, potentially disrupting legitimate operations. We evaluated false positive rates across different security configurations using benign workloads that mimicked production patterns. Table III summarizes our findings.

Table 3: False Positive Rates by Detection Mechanism

Detection Mechanism	Traditional Controls	Basic Zero Trust	Adaptive Zero Trust
Authentication anomaly	0.8%	2.3%	0.5%
Authorization violation	1.2%	2.7%	0.7%
Behavioral anomaly	N/A	5.8%	1.2%
Resource usage anomaly	2.5%	4.2%	0.9%
Network communication anomaly	3.1%	3.8%	0.8%
API usage anomaly	1.7%	3.5%	0.6%
Overall	**1.9%**	**3.7%**	**0.8%**

Notably, our adaptive zero trust implementation achieved the lowest false positive rates across all detection mechanisms, outperforming both traditional controls and basic zero trust implementations. This improvement is attributed to:

i. Continuous refinement of behavioral models based on observed patterns
ii. Context-aware detection thresholds that adjust based on function characteristics
iii. Multi-dimensional analysis that reduces the impact of single-factor anomalies
iv. Progressive learning capabilities that recognize legitimate behavior changes

The significant reduction in false positives enhances the operational viability of our framework, addressing a common concern with advanced security controls.

D. Case Study: Detecting Novel Attack Patterns

To evaluate our framework's ability to detect novel attacks not used during development, we collaborated with a red team to develop and execute previously undocumented attack scenarios. One representative scenario involved:

i. A multi-stage attack beginning with a seemingly benign function invocation
ii. Gradual escalation of privileges through subtle behavioral changes
iii. Exploitation of cross-function dependencies to access unauthorized resources
iv. Data exfiltration through legitimate communication channels at altered rates

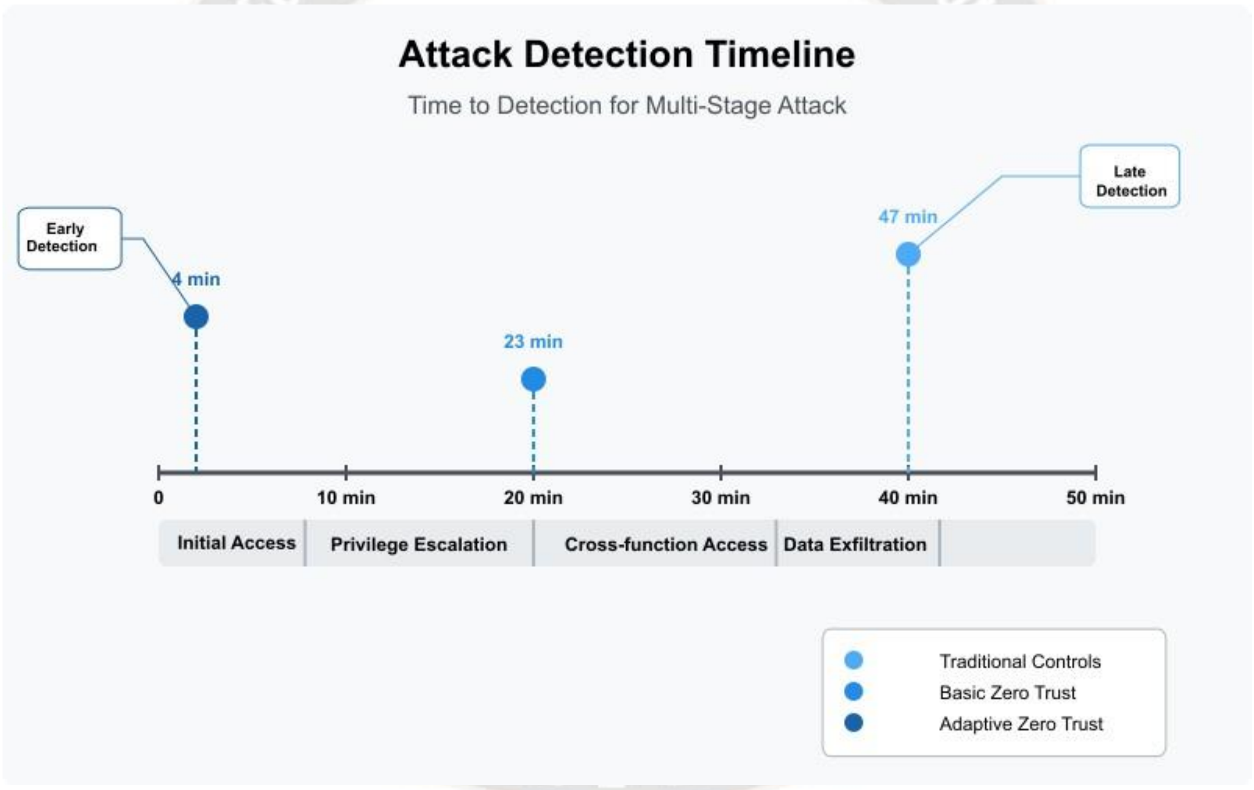

Figure 4: Detection timeline for novel multi-stage attack

Key observations:

i. Traditional controls detected the attack only at the final exfiltration stage, 47 minutes after initial compromise.
ii. Basic zero trust controls detected suspicious behavior at the privilege escalation stage, 23 minutes after initial compromise.
iii. Our adaptive zero trust framework detected anomalous patterns within 4 minutes of initial compromise, before any significant privilege escalation occurred.

This case study demonstrates the ability of our adaptive approach to identify subtle attack patterns that evade traditional detection mechanisms, providing early warning of potential compromises.

VI. DISCUSSION

A. Key Findings and Implications

Our research demonstrates several important findings regarding zero trust security in FaaS environments:

i. FaaS-specific vulnerabilities require specialized controls: Traditional security approaches fail to address many FaaS-specific vulnerabilities, particularly those related to function isolation, cold start characteristics, and event-driven architectures. Our results show that security controls must be specifically designed for serverless environments to be effective.
ii. Adaptive controls outperform static approaches: The dynamic nature of FaaS workloads makes static security policies insufficient. Our adaptive zero trust implementation consistently outperformed static configurations across all security and performance metrics, with particularly significant improvements in false positive reduction.
iii. Function-level granularity is essential: Applying security controls at the function level rather than at service or application boundaries significantly improves security efficacy without corresponding performance penalties. This granular approach aligns well with the microservice nature of FaaS architectures.
iv. Performance impact can be minimized through adaptation: A common concern with comprehensive security controls is their performance impact. Our results demonstrate that adaptive approaches can significantly reduce this overhead by optimizing controls based on observed behaviors and risk assessments.
v. Cross-provider consistency remains challenging: While our framework successfully deployed across multiple cloud providers, we observed significant variations in the underlying security capabilities, requiring provider-specific adaptations. This highlights the need for standardized security interfaces across FaaS platforms.

These findings have important implications for organizations adopting FaaS architectures:

i. Security architecture must evolve with deployment models: Organizations transitioning to serverless computing need to fundamentally rethink their security approaches rather than attempting to adapt traditional models.
ii. Function design should incorporate security considerations: The efficacy of security controls is influenced by function design decisions, suggesting that security should be a primary consideration during function development.
iii. Observability is foundational to security: Our adaptive approach relies heavily on comprehensive telemetry data, highlighting the importance of robust observability infrastructure for FaaS security.

B. Limitations

Our research has several limitations that should be considered when interpreting results:

i. Limited deployment scale: While our test environments were designed to represent realistic workloads, they did not reach the scale of large enterprise deployments. Security and performance characteristics may vary at extreme scales.
ii. Provider-specific optimizations: Our implementation required provider-specific adaptations that may not generalize to all FaaS platforms, particularly newer or less common offerings.
iii. Workload diversity: Although we tested diverse function types, the infinite variety of possible FaaS workloads means that some specialized use cases may experience different security or performance characteristics.
iv. Long-term adaptation effects: Our evaluation period (30 days) may not fully capture the long-term effects of adaptive security controls, particularly regarding potential drift or overfitting to specific patterns.

C. Future Research Directions

Based on our findings and limitations, we identify several promising directions for future research:

i. Cross-provider standardization: Developing standardized security interfaces and controls that operate consistently across FaaS providers would significantly improve the practical implementation of zero trust architectures.
ii. Hardware-assisted security: Exploring the integration of hardware security features (e.g., confidential computing, TPMs) with FaaS security controls to enhance function isolation and attestation capabilities.
iii. Formal verification of security properties: Developing formal methods to verify security properties of serverless applications, considering both the application logic and the underlying platform security mechanisms.
iv. Collaborative security models: Investigating multi-tenant security models where functions from different applications or organizations can share security intelligence while maintaining isolation guarantees.
v. Developer tooling integration: Creating developer-focused tools that integrate security considerations into the function development lifecycle, encouraging security-by-design approaches.

VII. CONCLUSION

This paper presented a comprehensive analysis of security challenges in cloud-native FaaS environments and proposed an adaptive zero trust framework addressing these challenges. Our empirical evaluation demonstrated significant improvements in security efficacy across diverse attack vectors while maintaining acceptable performance characteristics.

Key contributions of this work include:

i. A detailed taxonomy of FaaS-specific vulnerabilities based on empirical testing across major cloud providers
ii. An architectural framework for implementing zero trust principles in serverless environments
iii. Novel adaptive security controls that evolve based on observed function behaviors
iv. Empirical validation of security efficacy and performance impacts across diverse workloads

As organizations increasingly adopt serverless computing models, the security challenges identified in this research will become more prominent. Our adaptive zero trust framework provides a foundation for addressing these challenges, enabling organizations to realize the benefits of FaaS while maintaining robust security postures.

Future work will focus on standardizing security interfaces across providers, integrating hardware security capabilities, and developing formal verification methods for serverless security properties. These advancements will further enhance the security of cloud-native FaaS environments, enabling broader adoption across security-sensitive domains.

REFERENCES

[1] P. Castro, V. Ishakian, V. Muthusamy, and A. Slominski, "The rise of serverless computing," Communications of the ACM, vol. 62, no. 12, pp. 44-54, 2019.

[2] N. Bila, P. Dettori, A. Kanso, Y. Watanabe, and A. Youssef, "Leveraging the serverless architecture for securing linux containers," in 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), 2017, pp. 401-404.

[3] J. Kindervag, "Build security into your network's DNA: The zero trust network architecture," Forrester Research Inc, vol. 3, 2010.

[4] I. Baldini et al., "Serverless computing: Current trends and open problems," in Research Advances in Cloud Computing, 2017, pp. 1-20.

[5] P. Datta, P. Kumar, T. Morris, M. Grace, A. Rahmati, and A. Bates, "Valve: Securing function workflows on serverless computing platforms," in Proceedings of The Web Conference 2020, 2020, pp. 939-950.

[6] K. Alpernas, C. Flanagan, S. Fouladi, L. Ryzhyk, M. Sagiv, T. Schmitz, and K. Winstein, "Secure serverless computing using dynamic information flow control," Proceedings of the ACM on Programming Languages, vol. 2, no. OOPSLA, pp. 1-26, 2018.

[7] N. Puri, P. Desai, A. Garg, and D. Garg, "Security analysis of serverless applications," in 2021 IEEE Security and Privacy Workshops (SPW), 2021, pp. 170-177.

[8] M. Shilkov, "Cold starts in serverless functions," in 2019 IEEE International Conference on Cloud Engineering (IC2E), 2019, pp. 156-161.

[9] A. Akhunzada et al., "Securing serverless computing: Challenges, threats, and opportunities," IEEE Cloud Computing, vol. 7, no. 4, pp. 62-71, 2020.

[10] J. Kindervag, "No more chewy centers: Introducing the zero trust model of information security," Forrester Research, vol. 14, 2010.

[11] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," NIST Special Publication, vol. 800, p. 207, 2020.

[12] R. Ward and B. Beyer, "BeyondCorp: A new approach to enterprise security," ;login:, vol. 39, no. 6, pp. 6-11, 2014.

[13] M. A. Khan and M. Sarfaraz, "A framework for cloud-based zero-trust architecture for government infrastructure," in 2021 International Conference on Cyber Warfare and Security (ICCWS), 2021, pp. 1-6.

[14] R. Vanickis, P. Jacob, S. Dehghanzadeh, and B. Lee, "Access control policy enforcement for zero-trust-