UDC 004.056.55 **doi:** https://doi.org/10.20998/3083-6298.2025.02.06

Vladyslav Maksymov[1]

[1]Kharkiv National University of Radio Electronics, Kharkiv, Ukraine

ARCHITECTURAL PRINCIPLES AND OPERATIONAL PRACTICES FOR BUILDING SECURE DIGITAL INFRASTRUCTURE IN CLOUD ENVIRONMENTS

Abstract. Topicality. Internet ecosystems evolve faster than traditional enterprise lifecycles, which results in the constant emergence of new attack vectors and growing risks of data leakage, data loss, and SLA violations. Security is no longer limited to being a property of code; it has become an end-to-end attribute of the entire ecosystem, encompassing identities, networks, data, applications, processes, and telemetry. **Subject of study**. Multi-layered security for cloud infrastructures and web applications that combines Zero Trust, defense in depth, secrets management, privacy controls, DevSecOps practices, and correlation of logs, metrics, and traces. **Purpose.** To create a reproducible scaffold of architectural principles and operational practices that reduces the attack surface, shortens MTTD and MTTR, supports SLO and SLA compliance, and facilitates alignment with control catalogs such as CIS AWS Foundations and industry frameworks including NIST SP 800-207, NIST SP 800-53, ISO/IEC 27001, CSA CCM, and the OWASP Top Ten. Methods. Isolation of environments and trust boundaries; role-based access with MFA and short-lived credentials; centralized secrets management and rotation; private networks and micro-segmentation; pervasive encryption at rest and in transit; data lifecycle and privacy controls; implementation of security gates in CI/CD; standardized configuration baselines and continuous compliance scanning; centralized logging, distributed tracing, and guided incident response. **Results.** A detailed set of policies and sub-practices with clearly defined goals, procedures, artifacts, acceptance criteria, and metrics; generalized figures representing security posture; a table of operational targets; and an analysis of observability's role in improving MTTD and MTTR. **Conclusions.** The integration of security standards and observability into both system architecture and operational lifecycle improves system resilience, strengthens auditability, and ensures that risks remain manageable while maintaining acceptable operational costs.

Keywords: Cloud security; AWS security best practices; Zero Trust architecture; Defense in Depth; Data protection; Cybersecurity risk management; Secure software development lifecycle; Application security; Network segmentation; Encryption; Monitoring and surveillance; Incident response; Compliance.

Introduction

Problem relevance. Internet ecosystems are in constant flux, which accelerates threat evolution: phishing improves, supply-chain exploits become more intricate, and cloud misconfigurations are a growing attack vector. Production systems built with older paradigms often continue to operate, creating a gap between modern threats and legacy security practices.

The severity stems from data value: incidents entail not only direct financial losses but also long-term reputational damage, regulatory exposure, and loss of user trust. With massive cloud adoption, security must be engineered as a property of architecture, processes, and operations, not merely a code-level add-on.

We address end-to-end security in an AWS-like cloud context while keeping recommendations technology-neutral. The novelty is a coherent policy-and-process scaffold with measurable metrics covering identity administration, network segmentation, encryption and key management, privacy and data lifecycle, DevSecOps, observability, and incident response.

Literature review. Zero Trust (NIST SP 800-207) defines a departure from implicit trust and continuous verification of identity, context, and device posture. NIST SP 800-53 Rev.5 systematizes controls from identity and cryptography to logging and incident response. ISO/IEC 27001:2022 defines ISMS requirements grounded in risk management. The CSA Cloud Controls Matrix (CCM v4/v4.1) maps cloud control objectives and eases cross-framework alignment.

The CIS AWS Foundations Benchmark v1.4.0 offers prescriptive configuration checks across identities, logging, monitoring, and service baselines. At the application layer, we follow OWASP Top-10:2021 and the Cheat Sheet Series for microservices, secrets, REST, and authentication.

Empirical work shows that observability and DevSecOps practices reduce MTTR, accelerate root-cause localization, and decrease the frequency of production vulnerabilities, while acknowledging challenges of alert noise, telemetry storage cost, and configuration drift. This motivates a scaffold that couples controls, processes, and metrics to balance security, speed, and cost.

The purpose of the research is to develop a comprehensive, reproducible framework of architectural principles and operational practices for securing cloud infrastructures and web applications. The framework seeks to minimize the attack surface, improve resilience through reduced Mean Time to Detect (MTTD) and Mean Time to Recover (MTTR), ensure compliance with Service Level Objectives (SLO) and Service Level Agreements (SLA), and streamline alignment with internationally recognized security benchmarks, partner certification requirements, and industry best practices.

This work emphasizes a holistic approach to security that integrates governance, risk management, compliance, secure software engineering, and operational resilience into a unified model. The approach is intended to be adaptable to different cloud service providers, with a primary reference to Amazon Web

Services (AWS), while maintaining applicability to hybrid and multi-cloud environments.

Objectives:

1. Design a stratified security control model. Establish layered controls, trust boundaries, and defense in depth mechanisms that provide granular isolation between environments and workloads.

2. Define policies and standards for Identity and Access Management (IAM), secrets handling, network segmentation, and encryption (both in transit and at rest), ensuring adherence to the principle of least privilege and regulatory requirements.

3. Specify requirements for data lifecycle management and privacy protection, including data classification, retention, deletion policies, and compliance with frameworks such as GDPR and CCPA.

4. Integrate security principles into the Software Development Life Cycle (SDLC) and CI/CD pipelines, embedding secure coding standards, automated vulnerability scanning, and policy enforcement at all stages of delivery.

5. Architect observability, monitoring, and incident response processes. Develop a telemetry driven security operations capability using centralized logging, anomaly detection, and forensic readiness.

6. Define measurable performance indicators. Establish Key Performance Indicators (KPI), Service Level Indicators (SLI), and Service Level Objectives (SLO) alongside thresholds and alerting strategies to maintain operational and security performance.

7. Demonstrate the framework's impact on operational risk. Use case studies and simulated attack scenarios to illustrate measurable improvements in risk index scores, MTTD, and MTTR metrics.

1. Methods, Models, and Algorithmic Foundations

The methodological basis of this research is a hybrid integration of Zero Trust Architecture (ZTA) principles and Defense-in-Depth (DiD) strategies. This approach combines identity-centric and context-aware access control, fine-grained network micro-segmentation, pervasive encryption, centralized secrets management, and comprehensive observability across infrastructure and application layers. The objective is to create a unified, layered security framework that applies to cloud-native, hybrid, and multi-cloud environments.

At the conceptual level, the framework enforces:

•Identity and Context-Based Access Control (ICBAC): All access decisions are validated against dynamic contextual parameters (user identity, device state, geolocation, and behavioral patterns).

•Network Micro-Segmentation: Logical and physical isolation of workloads to prevent lateral movement within the infrastructure.

• Pervasive Encryption: End-to-end encryption in transit and at rest, leveraging managed key infrastructure and hardware security modules (HSM).

• Centralized Secrets Management: Secure storage, distribution, and rotation of credentials, keys, and tokens, with automated policy enforcement.

• Total Observability: Continuous collection and analysis of telemetry data, logs, traces, and metrics for anomaly detection, incident response, and forensic analysis.

From a quantitative perspective, the framework uses a composite set of metrics to evaluate security posture and operational resilience. These include:

•Risk Index $R \in [0,1]$: A normalized measure of overall exposure to threats, with 1 representing maximum risk

• Control Coverage $C \in [0,1]$: The fraction of relevant security controls that are fully implemented and verified.

• Mean Time to Detect (MTTD) and Mean Time to Recover (MTTR): Operational metrics for detection and recovery latency.

• Composite Security Score S

Mapping to Standards and Best Practices:

The defined controls and processes are aligned with the CIS AWS Foundations Benchmark and the AWS Well-Architected Framework (Security Pillar) for infrastructure-level practices. For application-level security, the model maps to OWASP Top 10 categories and associated OWASP Cheat Sheets, ensuring coverage of common vulnerability classes such as injection, broken authentication, and sensitive data exposure.

The proposed algorithmic approach enables repeatable measurement, auditability, and continuous improvement, supporting periodic compliance checks, automated drift detection, and adaptive policy enforcement in dynamic cloud environments.

2. Policies and Engineering Practices for a Secure Cloud

Purpose and scope. This section regulates mandatory and recommended security practices for cloud infrastructure and web applications with an emphasis on AWS-like environments. It targets architects, security engineers, SRE/DevOps, and developers. Applicability boundaries: enterprise systems processing personal or commercially sensitive data, multi-account landscapes, microservice topologies.

Terms and abbreviations. MFA refers to multi factor authentication. IAM refers to identity and access management. TLS refers to mutual transport layer security. KMS refers to key management service. SBOM refers to software bill of materials. SAST, DAST, and SCA refer to static analysis, dynamic analysis, and dependency analysis respectively. MTTD and MTTR refer to mean time to detect and mean time to recover. SLI, SLO, and SLA refer to service level indicator, service level objective, and service level agreement. DSR refers to data subject request. DR refers to disaster recovery.

Context and assumptions. Environments are separated (prod/stage/dev); managed cloud services are used; centralized logs, tracing, and CI/CD exist; risk management follows ISO/IEC 27001. Controls are aligned with NIST SP 800-207/-53, CSA CCM, CIS AWS Foundations, and OWASP. Limitations: security

does not replace change management, peer review, or business accountability for data classification.

Metrics and quality control. KPI/SLI/SLO: risk index R; control coverage C; MTTD, MTTR; MFA coverage; share of short-lived credentials; encryption and trace coverage; compliance score vs. benchmarks; DSR SLA; DR RPO/RTO. Recording: centralized telemetry; reporting cadence: weekly/monthly; policy audits: quarterly with retrospectives.

Compliance and risks. Frameworks: CIS AWS Foundations, AWS Well-Architected (Security), NIST SP 800-207/-53, ISO/IEC 27001, CSA CCM, OWASP. Risks: credential theft, misconfiguration, supply-chain CVEs in dependencies and base images, uncontrolled egress leading to exfiltration, failures and data loss. Mitigations: environment isolation and trust boundaries, least privilege, short-lived credentials, pervasive encryption, DevSecOps gates, observability, and trained IR.

2.1. Account Architecture and Trust Boundaries. In the context of modern cloud ecosystems, the architecture of accounts and the delineation of trust boundaries play a decisive role in ensuring systemic resilience, regulatory compliance, and effective incident containment. The core objective here is to achieve strict isolation of environments and operational domains, thereby reducing the blast radius of potential security incidents and enabling more precise auditing and accountability.

From a governance perspective, it is imperative that production, staging, and development environments are hosted in physically and logically separated accounts. Cross account trust relationships must be established exclusively through short lived, least privilege roles that are automatically revoked or rotated after a defined interval. Administrative responsibilities should be divided among separate security, operations, and compliance teams to minimize the risk of privilege escalation. In exceptional cases such as urgent diagnostics in a controlled environment, temporary trust relationships may be granted, but always with a clearly documented rationale and a predefined expiration date.

This segmentation strategy is not merely a matter of policy. It is a proactive measure to localize and contain the impact of potential breaches, to facilitate forensic investigation, and to ensure alignment with industry frameworks such as the CIS AWS Foundations Benchmark and the AWS Well Architected Framework Security Pillar. The procedural workflow typically involves the hierarchical definition of the organization, followed by account structuring and the mapping of environments. This process includes the separation of billing channels, the deployment of dedicated logging accounts, and the implementation of delegated roles for environment specific operations. On a quarterly basis, trust boundaries are reviewed and obsolete or unused trust configurations are systematically decommissioned.

The tangible artifacts produced during this process, such as a centralized registry of accounts and trust relationships, architectural diagrams of trust boundaries, and detailed review records, serve both as operational references and as evidence of compliance during security audits. Success in this domain is measured against explicit quality gates. No production resources may be shared across environments. The account registry must remain up to date. Quarterly trust reviews must be successfully passed. All temporary trusts must be fully documented, including their justification and closure.

To maintain continuous assurance, a defined set of metrics is tracked. This includes the total count and age distribution of active trust relationships, the average time required to revoke a trust, and the percentage of services isolated by environment, which should remain at or above ninety five percent. Policy as code frameworks, automated account inventory systems, and compliance snapshot tools are deployed to enforce these standards at scale, ensuring that deviations are promptly detected and remediated. Exceptions to these rules are permissible only under the explicit authorization of the security leadership, with every deviation logged, justified, and subject to post event review.

By adopting such a rigorously controlled account architecture, organizations not only reduce their operational attack surface but also gain a robust foundation for sustainable security governance, adaptive threat response, and demonstrable compliance with prevailing cloud security benchmarks.

2.2. Identity and Access Management (IAM). Identity and Access Management (IAM) is a key component of cloud infrastructure security, ensuring control over authentication, authorization, and accountability for user and service actions. The goal is to provide access only to those subjects who need it, for the minimum possible time, while fully eliminating long-lived human credentials.

Mandatory requirements include 100% use of multi-factor authentication (MFA) for all human accounts and a complete ban on static access keys. It is recommended to enforce short time-to-live (TTL) values for elevated privilege sessions and apply Permission Boundaries to limit the maximum privilege scope. In exceptional cases (“break-glass” access), temporary privilege escalation is allowed with mandatory auditing and automatic session termination.

These rules apply to all human access channels and to critical machine integrations. Exceptions are permitted only for service roles operating in trusted environments. The main rationale is to reduce the attack window in case of credential compromise and to prevent privilege escalation, in alignment with CIS and NIST recommendations.

The process includes issuing short-lived sessions through an Identity Provider (IdP), splitting administrative duties between teams, regularly reviewing IAM policies, and immediately revoking access upon role changes or employee termination.

Key artifacts include a catalog of roles and policies, authentication logs, and privilege escalation reports. Acceptance criteria: MFA coverage at 100%, no static access keys, and session TTL values within defined thresholds.

Metrics: MFA coverage rate, median session TTL, access deactivation time, and the ratio of allowed to denied authorization requests. Automation is supported through federation platforms, IAM analytics, and sign-in monitoring. Exceptions are documented in writing, time-boxed, logged in the tracking system, and approved by the security lead.

2.3. Secrets and Certificate Management. Secrets and certificate management is a critical aspect of securing modern cloud-based and distributed systems. The primary goal is to eliminate the presence of sensitive credentials such as API keys, passwords, and private keys from source code repositories, container images, or other static artifacts, and to ensure a managed, auditable lifecycle for all secrets, keys, and certificates.

Mandatory requirements include the use of a centralized secret management system as the single authoritative store for all sensitive credentials, and the enforcement of runtime injection for secret delivery to applications and services. This approach prevents credentials from being embedded in code or stored in persistent images. It is recommended to implement automatic rotation of secrets and certificates and to maintain a full audit trail of all access and read operations. In some cases, short-lived tokens or leases may be issued to further reduce the attack surface.

These controls apply to all environments, from development to production. Exceptions are permitted only for fully isolated sandbox environments that have no connectivity to production data or systems. Centralizing secret management reduces the potential exfiltration surface, while runtime injection aligns with best practices outlined by OWASP and CIS benchmarks, ensuring that secrets never persist outside secure, controlled channels.

The standard process begins with defining access segmentation, ensuring that read permissions are granted only to the components or individuals that strictly require them. Rotation policies must be defined for each secret type, along with rules for certificate lifetime management. Systems should trigger alerts as certificates approach expiration (time-to-expiry, TTE) and conduct routine rotation drills to verify operational readiness.

Artifacts generated from this process include a complete registry of secrets, documented rotation policies, detailed access and rotation logs, and an up-to-date inventory of certificates. Acceptance criteria include zero occurrences of secrets in code repositories or container images, full validity of all certificates in use, and successful completion of rotation drills within the defined operational window.

Key metrics include the average and maximum age of secrets, a minimum of 90% automated secret and certificate rotation coverage, zero recorded incidents of "secret found in code," and maintaining a defined TTE buffer for certificates to avoid unexpected expirations.

Automation and tooling typically involve the use of centralized secret storage platforms, key management services (KMS), public key infrastructure (PKI) systems, leak detection scanners, and dashboards for monitoring secret age and certificate validity.

All exceptions must clearly designate an owner and an expiration date, be logged in the security tracking system, and be reviewed during each sprint to ensure timely remediation and risk minimization.

2.4. Networks, Micro-segmentation, and Controlled Egress. Network segmentation and controlled egress are foundational elements of a secure cloud and microservices architecture. The primary goal is to minimize the attack surface by enforcing a privacy-by-default approach, ensuring that only explicitly approved communication paths are available, and that outbound traffic is subject to strict control.

Mandatory requirements include enforcing a deny-by-default policy for all east–west (service-to-service) traffic and allowing public endpoints only when strictly necessary. It is recommended to establish mandatory Transport Layer Security (TLS) encryption for all inter-service communications and to use outbound traffic allow-lists or egress gateways to control external connectivity.

The scope covers all environments where services interact with each other or initiate outbound connections to the Internet, including production, staging, and critical development environments. The rationale is that network micro-segmentation replaces broad, permissive rules with fine-grained service identity–based controls, significantly reducing the likelihood of lateral movement by attackers. Similarly, outbound traffic control mitigates data exfiltration risks and blocks command-and-control (C2) channels commonly used in advanced persistent threats (APTs).

The canonical process begins by defining and documenting network perimeters and categorizing workloads into perimeter, application, and data tiers. Access control lists (ACLs) and security group policies are then configured to enforce segmentation rules. TLS encryption is applied to all service-to-service communications, with automated verification where possible. Segmentation and policy rules are periodically tested using penetration testing tools or simulation frameworks to ensure continued enforcement and to identify misconfigurations.

Artifacts include up-to-date segmentation diagrams, complete rule lists with justifications, test reports from segmentation validation exercises, and an inventory of all public endpoints in the environment. Acceptance criteria include the complete absence of "any-any" rules, the presence of only minimal and justified public endpoints, and active TLS protection for internal communication paths.

Key metrics include the total number of public endpoints, the percentage of internal communication paths protected by TLS (target $\geq 80\%$), the number of blocked unauthorized egress attempts, and the average lead time for implementing approved policy changes.

Recommended tooling and automation components include network policy controllers, distributed denial-of-service (DDoS) protection services, automated egress inventory tools, and real-time network event analyzers capable of detecting anomalous traffic flows.

Exceptions are permitted only on a temporary basis, must have a defined end date, and require escalation to the environment owner and the security team. All exceptions must be documented, monitored, and reviewed for closure before their expiration date.

2.5. Encryption and Key Management. Encryption and key management form the backbone of data confidentiality and integrity in any secure cloud or distributed system. The primary goal is to ensure that sensitive data remains protected both at rest and in transit, while maintaining an operational model that does not introduce excessive complexity or latency.

Mandatory requirements include 100% encryption coverage for all sensitive data (both at rest and in transit) using industry-accepted algorithms and protocols. At-rest encryption must be enforced through managed encryption services or equivalent, while in-transit encryption must rely on modern TLS configurations. It is recommended to maintain a separation of duties between data and key ownership to reduce the likelihood of insider threats, and to schedule planned cryptographic key rotations to limit the impact of potential key compromise. Where feasible, hardware-based roots of trust (e.g., HSMs) may be incorporated to strengthen assurance levels.

The scope applies to all data storage systems (databases, object storage, file systems) and all transport channels that handle sensitive or regulated data. The rationale is that establishing dedicated key hierarchies per data classification category minimizes the blast radius in the event of compromise, while strict TLS validation prevents interception, replay attacks, and data tampering during transmission.

The canonical process begins with the design of a multi-tiered key hierarchy, where master keys are used to encrypt data encryption keys (DEKs) for specific datasets or services. Certificate issuance and revocation must be automated to avoid manual intervention delays. Regular validation of TLS versions and cipher suites ensures alignment with current security baselines. Planned key rotation exercises must be executed in a way that avoids downtime or data loss, and these tests should be documented and repeatable.

Artifacts include a centralized registry of all keys and certificates, rotation policy documents, cryptographic operation logs, and documented test protocols for rotation drills. Acceptance criteria require full encryption coverage, the maintenance of a time-to-expiry (TTE) buffer for all certificates to prevent unexpected expiration, and the successful execution of rotation tests without data corruption or service interruption.

Metrics include encryption coverage (target 100%), average and maximum cryptographic key age, the percentage of network traffic secured with modern TLS versions, and the number of TLS-related connection failures. These metrics provide direct visibility into the security posture of the system and help drive timely remediation.

Recommended tools and automation frameworks include key management services (KMS), public key infrastructure (PKI) solutions, automated certificate management platforms, and TLS verification utilities.

Exceptions are extremely limited and must be publicly tracked within the organization, have a defined closure date, and be approved by the security governance board. All exceptions must be reviewed prior to expiration and either remediated or formally extended with documented justification.

2.6. Data, Privacy, and Lifecycle. Effective data lifecycle management combined with strict privacy controls is essential for minimizing risks to personal and sensitive information while ensuring compliance with applicable legal and regulatory frameworks. The primary goal is to handle all data according to clearly defined classification, retention, residency, and access policies, ensuring that sensitive data remains both secure and auditable throughout its lifecycle.

Mandatory requirements include comprehensive data classification and minimization practices, ensuring that only necessary data is collected and retained. Data retention periods must be explicitly defined, documented, and enforced. Geographic residency requirements should be implemented where applicable, especially for data subject to jurisdictional restrictions such as GDPR or CCPA. All backups must be tested regularly to ensure recoverability, and access to sensitive data fields must be logged in detail for audit and compliance purposes.

The scope covers all systems, storage repositories, and data flows that handle personal or sensitive information, as well as all access log records associated with such data. The rationale is that implementing object level and field level access controls enforces the principle of “only access your own data,” which significantly reduces the risk of unauthorized exposure. Regular disaster recovery (DR) testing ensures that business continuity can be maintained even in the event of data loss or system compromise.

The canonical process involves creating a detailed data catalog that categorizes all datasets by sensitivity and applicable regulatory requirements. Retention and geo residency policies must be defined for each category, and access control mechanisms, whether Attribute Based Access Control (ABAC) or Role Based Access Control (RBAC), should be applied down to the field level where applicable. Data subject rights (DSR) tooling should be implemented to support regulatory requests such as data access, correction, and deletion. DR tests must be conducted to ensure recovery point objectives (RPO) and recovery time objectives (RTO) are met without compromising data integrity.

Artifacts include the data catalog, retention and geo residency policy documents, detailed access logs for sensitive fields, and records of DR testing outcomes. Acceptance criteria require at least 95 percent valid classification coverage, adherence to all defined geo residency requirements, and demonstrated ability to perform DR recoveries within established RPO and RTO targets.

Key metrics include the percentage of datasets with valid classification, DSR service level agreement (SLA) compliance rate, DR test success rate, and the count of sensitive field access events accompanied by a complete

audit trail. These metrics provide continuous insight into the maturity of data governance and operational readiness.

Recommended tools and automation capabilities include Data Loss Prevention (DLP) and Information Rights Management (IRM) systems, enterprise grade data catalogs, field level access enforcement mechanisms, and automated DSR request management platforms.

Exceptions are not permitted for production environments containing sensitive personal data. The only permissible deviations apply to anonymized datasets, which must be processed under a separate documented procedure with explicit review and approval by the data governance team.

2.7. Application Lifecycle and DevSecOps. The integration of security into every stage of the Software Development Life Cycle (SDLC) is a fundamental principle of modern application engineering. The primary goal is to ensure that security controls are applied consistently and effectively throughout development, testing, deployment, and maintenance, without introducing delays that would critically impact release schedules.

Mandatory requirements include the implementation of quality gates for Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), Software Composition Analysis (SCA), container image scanning, Infrastructure as Code (IaC) validation, and automated detection of hardcoded secrets. Policy as code approaches are strongly recommended to ensure that security rules are versioned, auditable, and automatically enforced. The generation and maintenance of a Software Bill of Materials (SBOM) is also recommended to improve visibility into dependencies and to facilitate faster responses to vulnerabilities. All build artifacts must be digitally signed, and signature verification must be performed at deployment time to guarantee supply chain integrity.

This policy applies to all code, infrastructure definitions, and configuration changes destined for production environments. The rationale for these measures is clear: implementing security gates early in the SDLC significantly reduces the cost and complexity of remediating vulnerabilities, while SBOM and artifact signing establish trust in the software supply chain and accelerate remediation of Common Vulnerabilities and Exposures (CVE).

The canonical process involves strict isolation of build and runtime environments to prevent contamination of production systems. All external dependencies must be sourced from verified repositories, and their versions should be pinned to prevent unverified updates. Base images should be updated regularly with the latest security patches. During the build process, the SBOM must be generated and stored in a secure repository. All artifacts are digitally signed, and their signatures are verified automatically during deployment to prevent tampering.

Artifacts generated during this process include security gate reports, CVE tracking lists, current SBOM files, and digital signature metadata confirming artifact provenance. Acceptance criteria require that no critical CVEs are present at the time of release, that the SBOM is up to date, and that all production artifacts have been signed and verified.

Key metrics include mean time to remediate critical CVEs, the percentage of builds passing all security gates, SBOM coverage percentage, the percentage of reproducible builds, and the policy as code compliance rate. These metrics provide objective evidence of the security posture and operational efficiency of the DevSecOps process.

Recommended tools and automation solutions include integrated SAST and DAST scanners, SCA tools for dependency analysis, container and IaC scanners, artifact signing frameworks, and policy enforcement controllers.

Exceptions to these rules are permitted only with documented approval from the security technical board, and all such exceptions must undergo a post implementation retrospective to identify root causes and prevent recurrence.

2.8. Observability and Incident Response (IR). Ensuring rapid detection and effective resolution of security incidents requires comprehensive observability combined with a structured incident response framework. The primary goal is to minimize the Mean Time to Detect (MTTD) and the Mean Time to Recover (MTTR) by providing complete visibility into system behavior, enabling correlation of telemetry data, and establishing well-trained operational procedures.

The policy mandates the collection and storage of centralized control and data plane logs across all environments. All logs must follow unified schemas and contain standardized timestamps to ensure accurate correlation. The use of correlation identifiers is strongly recommended, as it enables linking logs, metrics, and traces associated with a single transaction or event. Distributed tracing should be implemented to capture end-to-end execution flows, while risk-oriented alerting should prioritize incidents with the highest potential impact.

This framework applies to all operational environments, including non-production systems where at least minimal logging is required to ensure the ability to reproduce incidents. The rationale is straightforward: the integration of log, metric, and trace data under a unified correlation identifier significantly accelerates forensic analysis, while the combination of signature-based detection with behavioral anomaly analysis reduces the likelihood of missing novel threats.

The canonical process begins with the integration of log, metric, and trace streams into a central observability platform. Alert thresholds and priorities are calibrated based on historical data and evolving threat models. Incident response drills are conducted regularly according to documented runbooks to maintain operational readiness. The effectiveness of alerting mechanisms is continuously evaluated through the measurement of precision, recall, and ingestion latency.

Artifacts generated in this process include log schema catalogs, operational dashboards, incident response runbooks, and journals of completed drills.

Acceptance criteria require that all key security-relevant actions are sufficiently logged, that trace coverage meets or exceeds defined targets, and that alerting systems maintain precision and recall within predefined performance bounds. MTTD and MTTR must remain within target thresholds agreed upon by stakeholders.

Metrics used for evaluation include MTTD and MTTR, the proportion of events with complete log-metric-trace correlation, average log ingestion latency, alert precision and recall rates, and the percentage of monitored transactions covered by distributed tracing. These metrics provide a quantifiable basis for evaluating both the detection capabilities and operational responsiveness of the system.

Recommended tooling and automation include centralized log collection services, application performance monitoring (APM) and tracing frameworks, incident management systems, and security analytics platforms capable of advanced correlation and anomaly detection.

Exceptions to these requirements are strictly limited. Disabling production logging or alerting mechanisms is prohibited without explicit approval from the designated incident response lead. Any authorized outage must be time-limited, fully documented, and include a plan for service restoration and post-event review.

2.9. Resilience, Fault-Tolerance, and Availability. Ensuring the resilience, fault tolerance, and high availability of systems is essential for maintaining agreed Service Level Objectives (SLO) and Service Level Agreements (SLA) even under adverse conditions such as component failures, network partitions, or sudden workload peaks. The goal is to guarantee that critical services continue to operate within performance and reliability targets while preventing cascading failures across dependent components.

The policy requires redundancy across multiple availability zones or regions for all critical services. This geographic and infrastructural separation minimizes the risk of complete service loss due to localized outages. Autoscaling mechanisms are recommended to dynamically adjust capacity based on demand, ensuring that resources are neither underutilized during low traffic periods nor overwhelmed during traffic surges. Additional architectural patterns such as bulkheads, circuit breakers, and rate-limiting are recommended to isolate failures and control traffic flow. Disaster recovery (DR) plans must exist for all critical services and undergo regular, controlled testing to verify readiness.

This framework applies to all components and services with strict SLO requirements. For less critical systems, partial implementation is possible, but still within a structured degradation plan. The rationale is that designing for idempotency and incorporating message queues helps mitigate data duplication and loss during partial failures, while chaos engineering experiments can reveal weaknesses before real incidents occur.

The canonical process begins with the separation of stateful and stateless components, followed by the introduction of message queues to decouple services and absorb transient spikes. Continuous health checks, self-healing mechanisms, and automated failover procedures are implemented to ensure swift recovery from failures. Regular disaster recovery drills and controlled chaos experiments validate the operational readiness of the system and refine incident playbooks.

Artifacts generated in this process include SLO monitoring dashboards, error budget reports, detailed disaster recovery and chaos experiment protocols, and predefined service degradation plans. Acceptance criteria require that SLO targets are consistently met, that disaster recovery and chaos engineering tests are documented, and that corrective actions from these tests are implemented within agreed timelines.

Key metrics include SLO attainment rates, error budget burn rates, failover execution times, autoscaling reaction times, available capacity headroom, and the percentage of chaos experiments completed successfully. These metrics provide an evidence-based evaluation of the system's resilience posture and its ability to recover from faults without breaching contractual commitments.

Recommended tools and automation include container orchestrators with built-in health management, autoscaling frameworks, queue managers for asynchronous communication, and specialized platforms for chaos engineering.

Exceptions are not permitted for services forming part of business-critical workflows. For non-critical systems, exceptions may follow an approved degradation plan that ensures graceful service reduction rather than abrupt outages. All exceptions must be documented and subject to periodic review.

2.10. Configuration Baselines, Patch Management, and Compliance. The objective of configuration baselines, patch management, and compliance enforcement is to minimize the risk of misconfigurations, reduce the time between the discovery of vulnerabilities and their remediation, and ensure that all systems remain consistently aligned with organizational benchmarks and security policies. Maintaining standardized and verified configurations across all environments strengthens the security posture by reducing variability, which is a common source of operational and security failures.

The policy requires the use of standardized base images and configuration profiles for all systems, including production services, development environments, continuous integration and continuous delivery (CI/CD) pipelines, and administrative workstations. Continuous compliance scanning must be implemented to detect configuration drift in real time. Critical patches should be applied within a predefined Service Level Agreement (SLA) timeframe, and any exceptions to this process should be strictly managed, time-bound, and formally approved.

This framework applies to all infrastructure components and services regardless of their criticality. While some non-critical systems may receive delayed patching under controlled circumstances, all deviations from the baseline must be justified, documented, and monitored. The rationale is that unified configuration profiles greatly reduce the likelihood of human error,

while continuous compliance monitoring identifies and addresses deviations before they become exploitable vulnerabilities. This approach aligns with the recommendations of industry standards such as CIS Benchmarks, NIST guidelines, and the AWS Well-Architected Framework Security Pillar.

The canonical process begins with the definition and maintenance of baseline configurations that meet both security and operational requirements. These baselines are stored in version-controlled repositories to ensure traceability. Automated drift detection tools monitor all systems, triggering alerts or initiating remediation workflows when deviations are detected. Safe deviations can be remediated automatically, while high-impact changes require manual intervention. Exceptions must include an assigned owner and an expiration date, and evidence of remediation activities must be securely stored for auditing purposes.

Artifacts generated by this process include the configuration baseline profiles, the exception registry, compliance scan reports, and remediation evidence logs. Acceptance criteria include maintaining a compliance score above the defined threshold, ensuring that critical patches are deployed within the agreed SLA, and having no overdue exceptions in the registry.

Key metrics for monitoring this control include the mean time to apply patches, the overall compliance score, the rate of configuration drift, the count and average age of exceptions, and adherence to remediation SLAs. These metrics are vital for evaluating the efficiency and consistency of the configuration management process and for demonstrating compliance during external or internal audits.

Recommended tooling and automation include compliance scanning platforms, patch management solutions, policy-as-code frameworks for consistent enforcement, and automated evidence reporting systems for audit readiness.

Exceptions are only permitted when approved by the security technical board and must be recorded in a publicly accessible tracking system with a clearly defined closure date. Regular reviews of exception status are mandatory to ensure that deviations are resolved within their approved timeframes.

3. System Description, Figures, Experiments and Results

The proposed security architecture represents a stratified control system that integrates multiple, mutually reinforcing layers of defense. Its core structure encompasses the following domains: account-level trust boundaries, granular network micro-segmentation, Identity and Access Management (IAM) with short-lived credentials, centralized secrets management, pervasive encryption, DevSecOps gates integrated into the software delivery lifecycle, and an observability and incident response (IR) framework. These elements collectively form a “defense-in-depth” model designed to minimize the attack surface, reduce detection and recovery times, and ensure compliance with established security benchmarks.

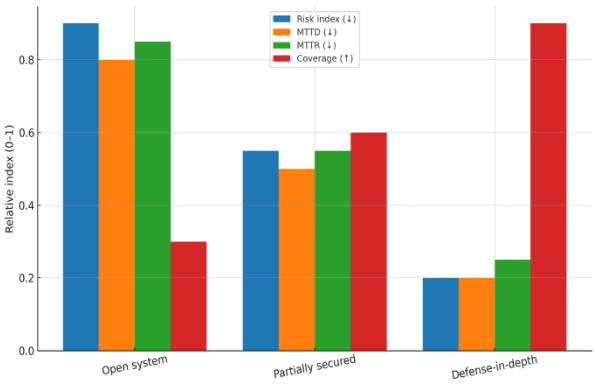

Fig.1. Security posture comparison

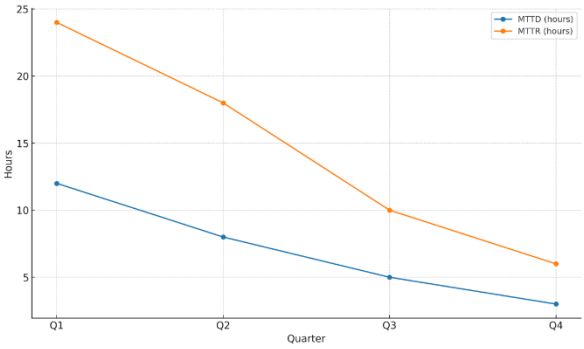

Fig.2. Quarterly trend driven by centralized logging/tracing and improved IR runbooks

Figure 1 illustrates a comparative analysis of security postures for three operational profiles: “open” systems with minimal controls, “partially secured” systems with selective implementation of safeguards, and “defense-in-depth” systems with comprehensive, integrated security measures. Key performance indicators include the risk index, Mean Time to Detect (MTTD), Mean Time to Recover (MTTR), and overall control coverage. The results clearly show that a defense-in-depth approach substantially reduces the risk index while improving detection and recovery times, as well as increasing the percentage of implemented security controls.

Figure 2 depicts a quarterly trend analysis of MTTD and MTTR improvements driven by the adoption of centralized logging, distributed tracing, and enhanced IR runbooks. The downward trend in both metrics over successive quarters demonstrates the tangible operational impact of continuous observability, precise correlation of telemetry, and rehearsed incident-handling procedures.

4. Discussion of results

The adoption of a comprehensive, multi-layered security framework inevitably introduces operational complexity. As security controls proliferate, the configuration surface expands, increasing the likelihood of misconfigurations if not rigorously governed. More granular controls and monitoring generate a larger volume of telemetry, which, while improving observability, also raises the risk of alert fatigue among operators. Excessive logging and tracing lead to higher storage costs, more complex indexing, and increased

computational overhead for analytics. Furthermore, stringent patch management cycles and compliance checks can delay feature releases or maintenance windows, especially in high-change environments.

This complexity has a measurable commercial impact. Each additional safeguard not only demands engineering resources for its implementation but also requires continuous investment in monitoring, auditing, and updating. For organizations with lean budgets or systems that handle minimal sensitive data, such investment can appear disproportionate to the perceived risk. In such contexts, a reduced or selectively applied control model may be sufficient. However, for platforms handling high-volume personal, financial, or mission-critical data, the benefits of comprehensive security overwhelmingly outweigh the drawbacks.

The advantages manifest across several dimensions. First, Mean Time to Detect (MTTD) and Mean Time to Recover (MTTR) typically decrease when telemetry correlation, automated incident response playbooks, and proactive anomaly detection are integrated. Second, the system's predictability and auditability improve, enabling faster and more confident responses to partner audits, certification reviews, and compliance assessments. Third, alignment with recognized benchmarks and standards (e.g., CIS AWS Foundations Benchmark, OWASP, NIST) increases stakeholder trust and market competitiveness.

Balancing these factors requires deliberate design choices. Risk-oriented logging ensures that high-value events are captured in detail, while low-impact noise is minimized through sampling and aggregation techniques. Data retention policies help control storage growth, ensuring that only the most relevant and legally required data is kept. Automated remediation for common misconfigurations or expired credentials reduces manual toil, lowers incident count, and supports faster recovery. Service Level Objective (SLO)-driven operations ensure that the degree of security enforcement aligns with the business's uptime and performance commitments.

A sustainable approach also requires cultural alignment between Security Operations (SecOps) and Site Reliability Engineering (SRE) teams. Cross functional collaboration ensures that security measures are integrated in a manner that does not undermine availability or performance objectives. Furthermore, incorporating structured post mortem reviews into the feedback loop ensures that the lessons learned from incidents are directly applied to the refinement of policies, controls, and automation scripts. This continuous improvement process reduces the probability of incident recurrence and contributes to the progressive maturity of the security program over time.

5. Conclusions

This study has presented a structured and internally consistent scaffold of security policies and operational practices, explicitly mapped to internationally recognized frameworks and standards including NIST, ISO 27001, Cloud Security Alliance (CSA), CIS Benchmarks, and the OWASP Top Ten. Each practice is supported by clearly defined processes, documented artifacts, quality gates, and measurable metrics, ensuring that implementation is both verifiable and repeatable across different organizational contexts.

The proposed layered security stratification, which includes account and trust boundary management, network micro-segmentation, identity and access controls, data governance, application-level protections, DevSecOps integration, and observability, significantly reduces the overall attack surface. This stratification also enhances auditability and regulatory compliance by creating discrete, independently verifiable security zones.

The integration of observability across logs, metrics, and distributed traces accelerates root-cause localization during incidents. By enabling high-fidelity correlation and anomaly detection, the approach demonstrably shortens Mean Time to Detect (MTTD) and Mean Time to Recover (MTTR), thereby directly supporting agreed Service Level Agreements (SLA) and improving operational resilience.

DevSecOps practices, including the implementation of security gates at multiple stages of the software delivery lifecycle, generation of Software Bills of Materials (SBOM), and cryptographic signing with deploy-time verification of artifacts, strengthen supply chain trust. These measures reduce the likelihood of production-level vulnerabilities being introduced through third-party dependencies, misconfigurations, or unverified build processes.

Recognizing the inherent trade-offs between security, operational complexity, and cost, the framework incorporates mitigating strategies such as automated remediation, risk-oriented telemetry collection, well-defined retention policies, and disciplined compliance tracking. These strategies help maintain an optimal balance between security assurance and operational efficiency, making the approach adaptable to both high-security and cost-sensitive environments.

References

1. Berardi, D., Giallorenzo, S., Mauro, J., Melis, A., Montesi, F. and Prandini, M. (2022), "Microservice security: a systematic literature review", *PeerJ Computer Science*, 8:e779, https://doi.org/10.7717/peerj-cs.779
2. Li, B., Peng, X., Xiang, Q. et al. (2022), "Enjoy your observability: an industrial survey of microservice tracing and analysis", *Empir Software Eng*, 27, 25, https://doi.org/10.1007/s10664-021-10063-9
3. Yeoh, W., Liu, M., Shore, M. and Jiang, F. (2023), "Zero trust cybersecurity: Critical success factors and A maturity assessment framework", https://doi.org/10.1016/j.cose.2023.103412
4. Chauhan, M. and Shiaeles, S. (2023), "An Analysis of Cloud Security Frameworks, Problems and Proposed Solutions Network", https://doi.org/10.3390/network3030018

5. Prates, L. and Pereira, R. (2025), "DevSecOps practices and tools", *Int. J. Inf. Secur*. 24, 11, https://doi.org/10.1007/s10207-024-00914-z
6. Wang, R., Li, C., Zhang, K. et al. (2025), "Zero-trust based dynamic access control for cloud computing", *Cybersecurity* 8, 12 https://doi.org/10.1186/s42400-024-00320-x
7. ICSE 2023 (2023), "An empirical study on Software Bill of Materials: where we stand and the road ahead", *In: Proceedings of ICSE*, https://doi.org/10.1109/ICSE48619.2023.00219
8. NIST SP 800-207 (2020), "Zero Trust Architecture"
9. NIST SP 800-53 Rev.5 (2020), "Security and Privacy Controls for Information Systems and Organizations"
10. Cloud Security Alliance (2024–2025), "Cloud Controls Matrix (CCM v4/v4.1) & CAIQ v4"
11. Center for Internet Security, "CIS AWS Foundations Benchmark v1.4.0"
12. OWASP Top-10:2021 (2021), "OWASP Cheat Sheet Series (REST, Secrets Management, Microservices Security)"

Received (Надійшла) 07.08.2025
Accepted for publication (Прийнята до друку) 26.08.2025

ВІДОМОСТІ ПРО АВТОРІВ/ ABOUT THE AUTHORS

Максимов Владислав Олександрович – інженер в галузі електроніки та телекомунікацій, спеціальність «Автоматизовані комплекси радіоелектронних виробництв», випускник Харківського національного університету радіоелектроніки, Харків, Україна;
Vladyslav Maksimov – Engineer in Electronics and Telecommunications, specializing in Automated Complexes of Radio-Electronic Productions, graduate of Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: maksimov.v.a.1990@gmail.com; ORCID Author ID: https://orcid.org/0009-0006-4555-8560.

АРХІТЕКТУРНІ ПРИНЦИПИ ТА ОПЕРАЦІЙНІ ПРАКТИКИ ПОБУДОВИ БЕЗПЕЧНОЇ ЦИФРОВОЇ ІНФРАСТРУКТУРИ В ХМАРНИХ СЕРЕДОВИЩАХ

В. О. Максимов

Анотація. Актуальність. Інтернет-екосистеми розвиваються швидше, ніж традиційні життєві цикли підприємств, що призводить до постійної появи нових векторів атак та зростання ризиків витоку даних, їх втрати та порушення рівня наданих послуг (SLA). Безпека більше не обмежується лише властивостями коду, а є наскрізною характеристикою всієї екосистеми, яка охоплює ідентичності, мережі, дані, додатки, процеси та телеметрію. **Предмет дослідження**. Багаторівнева безпека для хмарних інфраструктур та веб-застосунків, що поєднує підхід Zero Trust, концепцію «захист у глибину» (Defense in Depth), керування секретами, контроль конфіденційності, практики DevSecOps та кореляцію журналів, метрик і трасувань. **Мета.** Створення відтворюваного каркасу архітектурних принципів та операційних практик, що зменшують площину атаки, скорочують показники MTTD та MTTR, підтримують виконання SLO та SLA, а також сприяють узгодженню з контрольними каталогами, такими як CIS AWS Foundations, та галузевими стандартами, включно з NIST SP 800-207, NIST SP 800-53, ISO/IEC 27001, CSA CCM та OWASP Top Ten. Методи. Ізоляція середовищ та довірчих меж; рольовий доступ із багатофакторною автентифікацією та короткостроковими обліковими даними; централізоване керування секретами та їх ротація; приватні мережі та мікросегментація; повсюдне шифрування даних під час зберігання та передавання; контроль життєвого циклу та конфіденційності даних; впровадження контрольних точок безпеки в CI/CD; стандартизовані конфігураційні базові профілі та постійне сканування на відповідність вимогам; централізоване журналювання, розподілене трасування та керована реакція на інциденти. **Результати.** Розроблено детальний набір політик і підпрактик з чітко визначеними цілями, процедурами, артефактами, критеріями приймання та метриками; підготовлено узагальнені схеми, що відображають рівень безпеки; складено таблицю операційних цілей; виконано аналіз ролі спостережуваності у скороченні MTTD та MTTR. **Висновки.** Інтеграція стандартів безпеки та механізмів спостережуваності як у архітектуру системи, так і в операційний життєвий цикл підвищує стійкість системи, покращує можливості аудиту та забезпечує контрольованість ризиків при збереженні прийнятного рівня операційних витрат.

Ключові слова: безпека хмарних середовищ; практики безпеки AWS; архітектура Zero Trust; захист у глибину; захист даних; керування кіберризиками; безпечний життєвий цикл розробки ПЗ; безпека додатків; сегментація мережі; шифрування; моніторинг та спостереження; реагування на інциденти; відповідність вимогам.