



Zero Trust for SQL Server: A Three-Layer Security Architecture

MAYNARD I. CAPIL¹ , DANIEL D. DASIG JR.¹ 

¹De La Salle University Dasmariñas
City of Dasmariñas, Cavite, 4114, Philippines

Corresponding author: cmi2903@dlsud.edu.ph; dddasig@dlsud.edu.ph

Originality: 100% • Grammarly: 95% • Plagiarism: 0%

ABSTRACT

Article History

Received: 30 Jan 2026

Revised: 09 Aug 2026

Accepted: 13 Sept 2026

Published: 31 Oct 2026

Keywords— database security architecture, applied experimental research, SQL Server privilege management, Zero Trust security, Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC), Row-Level Security (RLS), Just-in-Time privilege elevation, least privilege enforcement, data classification

Enterprise database systems are frequently targeted due to their reliance on perimeter-based, static access control models that lack continuous verification and privilege minimization. This study designed and empirically validated a three-layer Zero Trust architecture for Microsoft SQL Server that natively integrates instance-level authentication governance, enhanced Role-Based Access Control (RBAC) with Row-Level Security (RLS) and Just-in-Time (JIT) privilege elevation, and metadata-driven Attribute-Based Access Control (ABAC) through data classification — all without requiring external security middleware. Employing design science and applied

experimental methodology, architecture was deployed in a live production environment comprising 589 databases and 132 identities over a 90-day post-implementation period. Chi-square tests of independence with Cramér's V



©Capil, M. I., & Dasig, D. D., Jr. (2026). Open Access. This article published by JPAIR Multidisciplinary Research is licensed under a Creative Commons Attribution-Noncommercial 4.0 International (CC BY-NC 4.0). You are free to share (copy and redistribute the material in any medium or format) and adapt (remix, transform, and build upon the material). Under the following terms, you must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. You may not use the material for commercial purposes. To view a copy of this license, visit: <https://creativecommons.org/licenses/by-nc/4.0/>

effect size confirmed statistically significant reductions: login misuse declined by 82.1%, malicious authentication attempts by 66.7%, deployment errors by 59.8%, and reporting disruptions by 43.7%. Overprivileged accounts decreased by 84.2%. These results demonstrate that a coordinated, database native Zero Trust pipeline substantially reduces attack surfaces and insider risk in enterprise SQL Server environments. The proposed architecture aligns with globally recognized compliance frameworks, including ISO/IEC 27001 and GDPR, offering a replicable and regulatory-ready deployment model for enterprise environments across diverse jurisdictions.

INTRODUCTION

The proliferation of multi-tenant, cloud-connected, and hybrid enterprise database environments has substantially expanded the attack surface of relational database management systems. Microsoft SQL Server, widely deployed across critical industries including finance, healthcare, and government, relies by default on broadly scoped built-in server roles such as `sysadmin` and `db_owner` that operate under implicit trust assumptions. These static configurations expose organizations to privilege misuse, insider threats, credential-based intrusions, and regulatory non-compliance with standards such as ISO/IEC 27001 and the General Data Protection Regulation (GDPR).

Traditional security frameworks applied to SQL Server environments have largely treated authentication governance, role management, and data classification as separate, independently administered controls. This fragmentation creates enforcement gaps: an authenticated user may escalate privileges through stale role assignments, excessive direct grants, or cross-database ownership chaining, even when perimeter-level controls are in place. The absence of a coordinated, database-native enforcement pipeline, one that continuously validates identity, restricts privileges to temporal necessity, and enforces data sensitivity at the row level — represents a critical gap in both academic research and enterprise practice.

The Zero Trust paradigm, formally articulated as “never trust, always verify,” has gained traction in network and identity security but has seen limited operationalization within database-layer enforcement (Guntaka, 2025; Mickie & Weng, 2025). Existing literature addresses RBAC scalability (Wu et al., 2020; Benedict, 2023), data flow security under role hierarchies (Logrippo, 2025), and ABAC’s contextual advantages over static role models (Harris, 2025), yet no prior work has demonstrated an integrated, production-validated architecture that unifies all three paradigms natively within SQL Server across a full session lifecycle.

This paper addresses that gap with three principal contributions: (1) a database-native Zero Trust implementation model for Microsoft SQL Server that eliminates reliance on external middleware by enforcing security controls directly within the engine's authentication, authorization, query execution, and session expiry layers; (2) a novel integration of temporal RBAC constraints and classification-based ABAC filtering, wherein JIT privilege elevation is governed by data sensitivity tiers — a combination not previously demonstrated in production-validated database security research; and (3) empirical validation of layered privilege control at enterprise scale, using a Chi-square test of independence and Cramér's V effect size analysis across 90-day pre- and post-implementation observation periods in a live multi-instance environment.

FRAMEWORK

This study is anchored on two foundational theoretical pillars: the Zero Trust security paradigm and layered access control theory. The Zero Trust paradigm, formally introduced by Kindervag (2010) and further operationalized by the National Institute of Standards and Technology (NIST SP 800-207), posits that no user, device, or network segment should be implicitly trusted, regardless of its location relative to a defined security perimeter. The principle of “never trust, always verify” underpins the entire enforcement logic of the proposed architecture. In the context of SQL Server, Zero Trust mandates continuous re-verification of identity and context at each phase of a session lifecycle from initial authentication through query execution rather than relying on a one-time perimeter check (Guntaka, 2025; Mickie & Weng, 2025).

Complementing Zero Trust is the layered access control framework, which synthesizes three complementary paradigms. Role-Based Access Control (RBAC) governs authorization by assigning permissions to job-function roles rather than to individual users, enabling scalable and auditable privilege management (Wu et al., 2020; Yonathan, 2025). Attribute-Based Access Control (ABAC) extends RBAC by incorporating contextual attributes such as department, clearance level, application name, client IP, and time of access — into authorization decisions, enabling fine-grained, sensitivity-aware access enforcement (Harris, 2025). The Principle of Least Privilege (PoLP), a foundational tenet of both paradigms, requires that each identity possess only the minimum permission necessary for its designated function at any given time (Carter, 2022; Wu et al., 2020).

The integration of these frameworks within a single, coordinated pipeline enforced natively within the SQL Server engine — constitutes the conceptual architecture of this study. Layer 1 operationalizes Zero Trust at the authentication

boundary; Layer 2 implements RBAC with temporal JIT constraints; and Layer 3 applies ABAC through metadata-driven data classification. The inter-layer dependencies ensure that trust is never assumed after initial login, that role membership is time-bound based on data sensitivity, and that session context is re-evaluated at query execution time. Together, these layers form a compounding, defence-in-depth security pipeline consistent with current enterprise security standards including ISO/IEC 27001.

OBJECTIVES OF THE STUDY

This study aims to design, implement, and empirically validate a three-layer Zero Trust security architecture natively within Microsoft SQL Server for a multi-tenant enterprise environment.

Specifically, the study seeks to: (1) design and implement a three-layer Zero Trust security architecture — encompassing instance-level authentication governance, enhanced RBAC with Row-Level Security and Just-in-Time access, and metadata-driven ABAC with data classification — entirely within the SQL Server engine without external middleware; (2) evaluate the architecture's measurable impact on security incidents, operational errors, and privilege exposure through pre- and post-implementation comparison over equivalent 90-day observation periods; and (3) establish a reproducible, statistically validated model applicable to enterprise SQL Server deployments of comparable scale and complexity.

LITERATURE REVIEW

The security limitations of default SQL Server configurations are well-documented. Wu et al. (2020) demonstrated that automated enforcement of the Principle of Least Privilege (PoLP) can significantly restrict excessive data-source permissions in large-scale RBAC deployments, establishing that programmatic role governance outperforms manual privilege assignment. Carter (2022) formalized a layered SQL Server security model spanning server, database, and object levels, emphasizing least privilege as a foundational constraint at each tier. Bush (2022) extended this work by integrating auditing specifications to support accountability and traceability of role activity, a mechanism directly applicable to the JIT logging requirements of the proposed architecture.

Database decomposition as a least-privilege strategy was explored by Sammartino and Baiardi (2024), who demonstrated that isolating sensitive data into granular database segments limits the blast radius of compromise.

Ingole et al. (2023) and Benedict (2023) confirmed that weak authentication schemes and poorly scoped privileges remain the dominant intrusion vectors in enterprise SQL Server environments, reinforcing the need for instance-boundary enforcement. The inadequacy of classical CIA-triad protections in the absence of granular privilege control has been specifically noted in the context of distributed and cloud-connected SQL Server deployments (Guntaka, 2025; Ulybyshev et al., 2023).

RBAC's administrative scalability continues to make it the dominant access control paradigm in enterprise environments (Yonathan, 2025; Benedict, 2023), and Logrippo (2025) formalized data-flow confidentiality within role hierarchies. However, static RBAC structures lack sensitivity-aware, instance-level enforcement — a limitation addressed by ABAC models that introduce contextual and resource attributes into authorization decisions (Harris, 2025). Steiner et al. (2018) and Sammartino and Baiardi (2024) demonstrated that hierarchical access control combined with elimination of redundant grants reduces breach propagation. SQL injection remains a persistent threat exacerbated by improper input validation and excessive role permissiveness (Rathi & Saxena, 2024; Hajar et al., 2024; Fadlil et al., 2024), reinforcing the case for row-level and classification-based filtering.

While Zero Trust principles — continuous verification and implicit trust elimination — have been articulated theoretically (Guntaka, 2025; Mickie & Weng, 2025), no prior production-validated study has demonstrated a unified SQL Server architecture combining instance-bound authentication governance, temporal RBAC constraints, and metadata-driven ABAC classification enforcement within a single, coordinated pipeline. This research directly addresses that gap.

METHODOLOGY

Research Design

A design science and applied experimental methodology was employed. The proposed three-layer Zero Trust framework was designed, implemented, and evaluated in a controlled production environment over a six-month period — four months of implementation followed by two comparative 90-day observation windows. Quantitative metrics were collected pre- and post-implementation and analyzed using inferential statistics.

Validation Environment

The architecture was deployed at an anonymized mid-sized, multi-tenant enterprise, operating a mix of Online Transaction Processing (OLTP) and analytical

workloads across three representative systems: a customer-facing CRM, an order-management system, and a financial reporting database. All company identifiers, tenant names, and sensitive configuration values were redacted; all reported results are aggregated and anonymized to preserve operational confidentiality. The environment comprised 589 SQL Server databases across multiple instances, 132 SQL logins including Active Directory groups and users, native SQL logins, and application accounts — many carrying excessive privileges — along with monitoring tools including custom Extended Events sessions, SQL Profiler, and SQL Server Audit for permissions tracking.

Over the 90-day baseline observation period, the following categories of incidents were documented, as summarized in Table 1.

Table 1

Documented Security and Operational Incidents During the 90-Day Pre-Implementation Baseline Period

Issue Type	Count	Description
Malicious traffic attempts	18	Unauthorized access attempts detected
Production deployment errors	102	Accidental schema alterations
Ad-hoc reporting disruptions	87	Database blocking and long-running queries
Misuse of SQL logins	78	Accidental data deletion and credential sharing

Note. ISO/IEC 27001 non-compliance was also documented due to insufficient separation of duties and absence of least-privilege enforcement.

Participants and Locale

The study involved 132 SQL identities across multiple departments including Finance, HR, Operations, Sales, CRM, Marketing, and Reporting, operating within an anonymized enterprise organization. Engagement with departmental stakeholders, the CTO, and a Security Administrator was conducted to govern data classification decisions and role design.

Metrics Collection

Login activity was captured using SQL Server Error Logs and monitoring alerts for all login attempts, including successful, failed, and blocked connections. Query performance was tracked through Extended Events and SQL Profiler for long-running queries, blocking chains, and deadlock events. Deployment errors were identified by analyzing release notifications and database change logs to detect unauthorized schema alterations or accidental deletions.

Data Validation

Cross-verification was performed by correlating SQL logs, Extended Events traces, and deployment logs with system monitoring tool outputs to ensure completeness and accuracy. Random sampling of login attempts and query execution data confirmed data consistency. All related database objects, triggers, policies, and ABAC rules were scripted to allow replication in other SQL Server environments.

Ethical Considerations

This study was conducted in full compliance with institutional ethical standards for research involving organizational data. Informed consent was obtained from Company A's executive leadership and IT governance stakeholders prior to deployment. All individual user identifiers, tenant names, and sensitive configuration details were anonymized or redacted in reported findings to preserve confidentiality and protect participant privacy. The research did not involve collection of personal health information or direct human subject participation beyond professional IT governance roles. No Institutional Review Board (IRB) determination was required given the non-interventional, systems-level nature of the study; however, the study adhered to principles of beneficence, data minimization, and transparency in reporting. The authors declare no conflict of interest arising from the deployment site or its affiliated entities.

AI policy

The authors hereby declare that Microsoft Copilot was utilized in the preparation of this manuscript solely for the purposes of formatting assistance and language refinement. Specifically, Microsoft Copilot was employed to improve sentence clarity, grammatical accuracy, and stylistic consistency in the presentation of the authors' original ideas and findings. At no point was Microsoft Copilot used to generate substantive content, produce data, construct arguments, develop the theoretical framework, or contribute to any intellectual component of the research. All scholarly contributions including conceptualization, methodology, data collection, analysis, interpretation, and conclusions remain entirely the original work of the authors. The authors assume full accountability for the integrity, accuracy, and originality of all content presented in this manuscript.

Architecture and Components

The proposed architecture operationalizes Zero Trust principles natively within SQL Server by coordinating enforcement across four sequential phases: (1) Authentication Phase — connection establishment with instance-boundary

validation; (2) Authorization Phase — role resolution with elimination of direct grants; (3) Query Execution Phase — Row-Level Security predicate evaluation against classification metadata; and (4) Session Lifecycle Phase — JIT expiration and automated privilege revocation. Figure 1 illustrates the complete three-layer enforcement pipeline and the inter-layer integration logic.



Figure 1. *Three-Layer Zero Trust Enforcement Pipeline for Microsoft SQL Server. The diagram illustrates the sequential enforcement pipeline spanning authentication (Layer 1), authorization and session lifecycle (Layer 2), and query execution (Layer 3), with inter-layer integration logic and a unified audit trail.*

Novel Integration Logic Across Layers

A distinguishing contribution of this architecture is the bidirectional interdependence of the three layers — a design that prior works treating RBAC, ABAC, and instance-level controls as independent mechanisms have not demonstrated. Layer 1 feeds Layer 2: the LOGON trigger evaluates `HOST_NAME()`, `APP_NAME()`, `ORIGINAL_LOGIN()`, and client IP before any role resolution occurs; an identity rejected at Layer 1 never reaches the RBAC engine, eliminating privilege escalation through valid but untrusted sessions. Layer 2 feeds Layer 3: role assignments are time-bound via JIT activation windows, and the ABAC predicate evaluates not only department and clearance level but also the temporal validity of the role — access to Level 3 and Level 4 data is denied if the JIT window has expired, even if the role remains technically assigned in the system catalog. Layer 3 reinforces Layer 1: session context attributes evaluated at the ABAC layer — application name, hostname, client IP, and business-hours validation — mirror and deepen the authentication criteria established at Layer 1, creating dual enforcement in which contextual trust is re-verified at query execution time. All three layers write to a common SQL Server Audit trail, enabling cross-layer incident correlation that would be invisible to single-layer audit systems.

Differentiation from Existing Approaches

Existing SQL Server security implementations typically apply one of three paradigms independently: instance-level hardening through login restrictions and auditing (Carter, 2022; Bush, 2022); RBAC restructuring with least-privilege role design (Wu et al., 2020; Yonathan, 2025); or data classification for regulatory compliance, often managed through external tools (Sammartino & Baiardi, 2024). The proposed architecture departs from these in three specific ways. First, it introduces temporal RBAC constraints directly coupled to data sensitivity tiers — JIT access window durations and approval requirements are determined by the classification level of data being accessed. Second, it achieves full engine-native enforcement without external identity providers, security middleware, or third-party tools, making it deployable in air-gapped and on-premises environments. Third, it provides empirical, production-scale validation at a scope — 589 databases, 132 identities, 90-day observation — that substantially exceeds prior conceptual or laboratory-scale demonstrations.

Technical Implementation

Layer 1 — Instance-Level Security. This layer enforces authentication governance at the SQL Server boundary. A server-level LOGON trigger validates

HOST_NAME(), APP_NAME(), ORIGINAL_LOGIN(), and client IP from EVENTDATA(). Connections from unregistered hostnames, untrusted IP ranges, or blacklisted application sources are terminated before database-level authorization is evaluated. Table 2 summarizes the data sources used at this layer, and Table 3 presents the login event classification model.

Table 2

Data Sources Used for Instance-Level Authentication Validation in Layer 1

Data Source	Purpose
SQL Server Audit	Login tracking
sys.dm_exec_connections	IP tracing
Extended Events	Real-time login telemetry
Error Logs	Failed authentication

Table 3

Login Event Classification Model for LOGON Trigger Enforcement at the SQL Server Boundary

Category	Description
Type I	Invalid credentials
Type II	Unauthorized IP/Host
Type III	Disabled account
Type IV	Privilege escalation attempt
Type V	Brute-force pattern

Layer 2 — Enhanced RBAC with Row-Level Security and JIT Access. All 132 SQL logins were audited for direct database and server permissions. Explicit grants were revoked and 98 logins were migrated to custom role assignments. Example roles include Finance_Read (read-only access to financial tables in the Data Warehouse, replacing prior db_owner assignment) and HR_Write (scoped write privileges for HR analysts performing bulk imports into the ATS). Row-Level Security predicates were applied to sensitive tables, ensuring users access only rows consistent with their role scope. JIT Access was implemented using time-bound role activation windows — access granted for predefined periods (e.g., four hours), automatically revoked by a SQL Agent job upon expiration, and fully logged in SQL Server Audit.

Layer 3 — Attribute-Based Access Control with Data Classification. All database objects were classified into four sensitivity tiers through collaboration

with departmental users, the CTO, and the Security Administrator, using SQL Server extended properties and metadata tagging to ensure metadata-driven enforcement. Table 4 presents the classification framework, Table 5 reports coverage results, and Table 6 lists the ABAC data sources.

Table 4
Four-Tier Data Sensitivity Classification Framework Used for Metadata-Driven ABAC Enforcement

Classification Level	Description	Example
Level 1 — Public	Non-sensitive operational data	Lookup tables
Level 2 — Internal	Business operational data	Sales aggregates
Level 3 — Confidential	Customer data	PII records
Level 4 — Restricted	Financial and regulatory data	Payroll, audit logs

Table 5
Database Object Classification Coverage Results Across All Sensitivity Tiers

Category	Total Objects	Tagged with Classification	% Tagged
HRIS	92	92	100.0%
Sales	108	105	97.2%
CRM	76	70	92.1%
Operations	84	80	95.2%
Marketing	55	50	90.9%
Reporting	174	160	92.0%
Total	589	557	94.6%

Table 6
Attribute-Based Access Control Data Sources Governing Contextual Authorization in Layer 3

Source	Purpose
SQL Server Audit	Access tracking
sys.fn_builtin_permissions	Permission validation
Extended Events	Real-time predicate evaluation
Metadata queries	Classification verification

Statistical Analysis

To determine whether observed reductions in security and operational incidents were statistically associated with deployment of the three-layer architecture, a Chi-square test of independence (χ^2) was applied. Incident frequencies were compared across two equivalent 90-day observation periods: a baseline period (pre-implementation) and a post-implementation period. For each incident category, a 2×2 contingency table was constructed comparing implementation status (Before vs. After) against incident occurrence (Observed vs. Not Observed). Chi-square statistics were computed as: $\chi^2 = \sum [(O - E)^2 / E]$. Effect size was assessed using Cramér's V: $V = \sqrt{[\chi^2 / N(k - 1)]}$, where N represents total observations and k represents the smaller dimension of the contingency table. Statistical significance was evaluated at a 95% confidence level ($\alpha = 0.05$) with one degree of freedom for all 2×2 tables. The null hypothesis (H_0) states that incident frequency is independent of framework deployment. The alternative hypothesis (H_1) states that incident frequency is associated with deployment of the layered security framework. Validity assumptions satisfied include: equal observation periods (90 days each), mutually exclusive incident classifications, independent observations across periods, and expected cell frequencies exceeding the minimum threshold of five.

SQL Server-Specific Threat Model

The evaluated system is a multi-instance SQL Server deployment operating in mixed authentication mode, functioning as a centralized trust boundary hosting classified data at Levels 1 through 4. The threat model considers two adversary profiles. The external actor possesses network reachability to the SQL Server TCP endpoint and the ability to initiate login attempts and submit arbitrary SQL queries upon authentication, but no OS-level or TLS compromise capability. The insider or compromised identity holds valid login credentials and role membership and can execute dynamic SQL within granted scope. Primary attack surfaces addressed include brute-force and password-spraying against SQL logins, mixed-mode authentication exploitation, context spoofing via APP_NAME() and HOST_NAME(), over-permissive db_owner or sysadmin assignments, EXECUTE AS and ownership chaining abuse, cross-database privilege propagation, injection-based data exfiltration, standing privilege persistence, and stale or orphaned logins.

Internal Consistency Validation Procedures for ABAC Classification Tagging and RBAC Role

For Classification Tagging (Layer 3 — ABAC): Internal consistency in data classification was ensured through a multi-stakeholder validation process. Initial sensitivity tier assignments were independently reviewed by the Security Administrator before being finalized. Cross-referencing was performed against existing data governance documentation and departmental data ownership records to verify that classification labels accurately reflected the actual sensitivity of each database object. Any discrepancies identified during review were resolved through structured reconciliation sessions prior to metadata tagging implementation.

For Role Migration (Layer 2 — RBAC): Role migration consistency was verified through a pre- and post-migration permission audit. Each of the 132 SQL logins was mapped against its prior direct grants and evaluated against the principle of least privilege before role assignment was finalized. Post-migration, automated queries were run against system catalog views to confirm that no residual direct grants remained and that role-to-login mappings were accurate and complete. A secondary review by the Security Administrator served as an additional consistency check before the architecture was deployed in the live production environment.

RESULTS AND DISCUSSION

During the 90-day post-implementation observation period, measurable and statistically significant improvements were recorded across all incident categories following deployment of the three-layer security framework, as summarized in Table 7.

Table 7
Post-Implementation Incident Frequency Results Across All Security Categories Following 90-Day Deployment

Incident Category	Baseline (90 days)	Post-Implementation (90 days)	Reduction (%)
Malicious login attempts	18	6	66.7%
Production deployment errors	102	41	59.8%
Ad-hoc reporting disruptions	87	49	43.7%
Misuse of SQL logins	78	14	82.1%
Over-privileged accounts	114	18	84.2%

Note. All reductions are statistically significant at $\alpha = 0.05$ (Chi-square test of independence, $df = 1$). Effect sizes assessed via Cramér's V indicate medium to large associations across all categories.

Malicious login attempts declined from 18 baseline cases to 6 post-implementations, a 66.7% reduction. This improvement is primarily attributed to LOGON trigger-enforced IP restrictions and automated failed-login lockout policies. While attempted connections from unapproved sources continued to be observed, the instance-boundary enforcement layer intercepted and blocked these attempts before any authentication token was issued, eliminating the risk of credential stuffing reaching the role resolution phase.

Production deployment errors declined from 102 to 41 cases, a 59.8% reduction. Prior to Layer 2 restructuring, widely assigned db_owner and sysadmin roles permitted uncontrolled schema modifications. Transition to custom job-function roles and JIT-governed elevated access required controlled approval and time-bound permissions for schema modification operations, substantially reducing unauthorized and accidental changes. Residual errors were concentrated in legacy integration pathways pending migration.

Ad-hoc reporting disruptions declined from 87 to 49 incidents, a 43.7% reduction. The introduction of role-based reporting views, RLS filtering, and workload governance through scoped permissions reduced contention and improved workload isolation. The moderate residual rate reflects continued load-related variability inherent to analytical workload patterns rather than access control deficiencies.

Misuse of SQL logins declined from 78 documented cases to 14, an 82.1% reduction. Post-implementation enforcement of role-only permissions, automated JIT expiration, and login review cycles substantially curtailed credential sharing, stale accounts, and cross-department data access. Residual cases primarily involved misconfigured service accounts and legacy integrations awaiting remediation.

Overall, over-privileged accounts were reduced from 114 to 18 users, an 84.2% reduction, substantially decreasing the attack surface and insider risk exposure.

Chi-square analysis confirmed statistically significant associations between framework deployment and incident reduction across all categories ($p < 0.05$, $\alpha = 0.05$, $df = 1$). The results demonstrate that the layered architecture produces cumulative security improvements exceeding what any single-layer control could achieve independently. Instance-level enforcement reduced external threat exposure by eliminating untrusted connections before role resolution. RBAC restructuring mitigated privilege abuse by removing excessive direct grants and introducing temporal constraints. ABAC with data classification prevented unauthorized cross-domain access at the row and object levels. The interdependence of these layers — wherein Layer 1 gates Layer 2, Layer

2 conditions Layer 3, and all three contribute to a unified audit pipeline — produced a compounding security effect that is the defining characteristic of the proposed architecture compared to prior independent-control implementations documented in the literature.

Despite the significant reductions observed across all incident categories, a subset of residual cases persisted primarily due to legacy system integrations that could not be immediately migrated to the new role-based framework without risking operational disruption. Specifically, several service accounts associated with third-party enterprise applications retained broader permission scopes than the principle of least privilege would otherwise permit, as these applications required elevated access for batch processing and cross-database reporting functions. These accounts were flagged for phased remediation but remained outside the full enforcement scope of the three-layer architecture during the observation period. This limitation acknowledges that database native Zero Trust implementation is most effective in greenfield or fully auditable environments, and that organizations with substantial legacy infrastructure may encounter transitional constraints that temporarily sustain residual risk. Future work should investigate compatibility frameworks for applying JIT and ABAC controls to legacy-dependent service accounts without compromising system continuity.

CONCLUSION

This study presented, implemented, and empirically validated a three-layer Zero Trust architecture for Microsoft SQL Server that uniquely integrates instance-level authentication governance, enhanced RBAC with Row-Level Security and Just-in-Time privilege elevation, and metadata-driven ABAC with data classification — entirely within the SQL Server engine, without external security middleware. The architecture was validated in a production environment of 589 databases and 132 identities over a 90-day post-implementation period, demonstrating statistically significant reductions in login misuse (82.1%), malicious authentication attempts (66.7%), deployment errors (59.8%), reporting disruptions (43.7%), and over-privileged accounts (84.2%).

The principal theoretical contribution is the formalization of a database-native Zero Trust enforcement model in which the three security layers function as an interdependent pipeline rather than independent controls — a design that addresses the research gap identified in the literature. The principal practical contribution is a fully scripted, reproducible implementation model applicable to comparable multi-instance SQL Server environments.

Limitations of this study include its reliance on a single anonymized

organization, which constrains the generalizability of specific quantitative findings. The deployment was conducted in a primarily on-premises SQL Server environment; applicability to Azure SQL Managed Instance and hybrid deployments warrants further investigation. Additionally, the 90-day evaluation window does not capture long-term behavioral adaptation by system users.

Future research directions include: (1) extension of the architecture to Azure SQL and hybrid cloud SQL Server deployments, incorporating cloud-native identity providers; (2) integration of machine learning-based anomaly detection to augment the rule-based LOGON trigger with adaptive behavioral baselines; (3) longitudinal evaluation beyond 90 days to assess the durability of privilege reduction and behavioral compliance; and (4) application of the framework to other relational database engines — Oracle Database and PostgreSQL — to assess cross-platform generalizability.

The proposed architecture's alignment with ISO/IEC 27001 access control requirements and GDPR data minimization principles confirms its regulatory readiness across diverse compliance jurisdictions. The database-native design requiring no external middleware further ensures that the model is replicable across organizations of varying infrastructure maturity, if SQL Server serves as the primary database engine. Looking forward, architecture's enforcement logic particularly the LOGON trigger validation, JIT privilege elevation, and metadata-driven ABAC present a viable foundation for extension into cloud-native deployments, including Azure SQL Managed Instance and hybrid SQL Server environments, where Zero Trust principles are increasingly mandated by cloud governance frameworks.

TRANSLATIONAL RESEARCH

The findings of this study can be translated into practice through multiple dissemination and application pathways. In terms of academic dissemination, the validated three-layer Zero Trust architecture can be published in peer-reviewed international journals and presented at information security and database management conferences, contributing to the scholarly discourse on database-native security frameworks and advancing Zero Trust operationalization research.

In enterprise IT governance and policy drafting, the architecture's enforcement pipeline — spanning authentication governance, least-privilege role management, and sensitivity-driven access filtering provides a concrete, implementable reference model for organizations developing or revising internal database security policies. IT auditors may directly adopt the pre- and post-implementation incident classification framework as a structured audit protocol

for evaluating privilege management maturity and access control effectiveness in SQL Server environments.

In professional training and certification contexts, the layered architecture offers a pedagogically coherent case study for IT security practitioners, database administrators, and compliance officers. The sequential enforcement model from instance-level authentication through JIT privilege elevation to metadata-driven ABAC maps naturally onto competency frameworks used in security training programs, including those aligned with ISACA's CISA and CISM certifications and (ISC)²'s CISSP domains.

In academic instructional design, the study's design science methodology, empirical validation approach, and statistical analysis framework provide a replicable template for graduate-level coursework in information security, database administration, and enterprise IT governance. Institutions offering programs in cybersecurity or information technology management may integrate the architecture as a practical case study illustrating the operationalization of Zero Trust principles within relational database systems.

ACKNOWLEDGMENT

The authors wish to thank the management and IT governance team of the anonymized organization (Company A) for their cooperation in providing production access, stakeholder engagement, and data governance support throughout the implementation and evaluation phases of this research.

Author Contributions: *Maynard I. Capil*—Conceptualization, Methodology, Data Curation, Formal Analysis, Writing – Original Draft; *Daniel D. Dasig*—Supervision, Wiring - Review

Funding: This Research Received No External Funding.

Institutional Review Board (Irb) Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: No New Data Were Created.

Conflicts of Interest: The Author Declares No Conflict of Interest.

LITERATURE CITED

- Benedict, J. O. S. E. P. H. (2023). An Appraisal of Database Security in a Business Organization (Case Study: Fintrak Software Company Limited). *University of East London, United kingdom*. <https://tinyurl.com/mrxapuft>
- Bush, J. (2022). Practical database auditing for Microsoft SQL Server and Azure SQL: Troubleshooting, regulatory compliance, and governance. Apress. <https://doi.org/10.1007/978-1-4842-8634-0>
- Carter, P. A. (2022). SQL Server Security Model. In *Pro SQL Server 2019 Administration: A Guide for the Modern DBA* (pp. 329-369). Berkeley, CA: Apress. https://doi.org/10.1007/978-1-4842-8864-1_10
- Fadlil, A., Riadi, I., & Mumin, M. A. (2024). Mitigation from SQL Injection Attacks on Web Server using Open Web Application Security Project Framework. <https://tinyurl.com/yc3c6x72>
- Guntaka, N. Y. R. (2025). Zero Trust and cloud identity: Building a resilient security framework. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 11(1). <https://doi.org/10.32628/CSEIT251112368>
- Hajar, S., Jaafar, A. G., & Rahim, F. A. (2024). A review of penetration testing process for SQL injection attack. *Open International Journal of Informatics*, 12(1), 72-87. DOI: <https://doi.org/10.11113/oiji2023.11n2.256>
- Harris, L. (2025). Role-Based Access Control (RBAC) vs. Attribute-Based Access Control (ABAC) in database systems. *ResearchGate*. <https://www.researchgate.net/publication/393429045>
- Ingole, K. R., Hage, A. S., Dudhabade, K. V., Tayade, S. D., Khewalkar, R. S., & Deshpande, S. N. (2023). Database security. *International Journal for Research in Applied Science & Engineering Technology*, 11(4). <https://doi.org/10.22214/ijraset.2023.50415>
- Logrippio, L. (2025). Data flow security in Role-based access control. *Journal of Information Security and Applications*, 90, 103997. <https://doi.org/10.1016/j.jisa.2025.103997>

- Mickie, J., & Weng, J. (2025). Zero Trust Security: A Proactive Cybersecurity Model for Risk Management. *Unpublished*. [https://doi.org/10.13140/RG.2\(20335.55201\)](https://doi.org/10.13140/RG.2.20335.55201). <https://doi.org/10.13140/RG.2.20335.55201>
- Rathi, K., & Saxena, D. (2024). SQL injection attack techniques and prevention [Conference proceedings].
- Sammartino, V., & Baiardi, F. (2024). Database Decomposition to satisfy the Least Privilege Principle in Healthcare. *ARIS2-Advanced Research on Information Systems Security*, 4(1), 47-69. <https://doi.org/10.56394/aris2.v4i1.43>
- Steiner, S., de Leon, D. C., & Jillepalli, A. A. (2018, April). Hardening web applications using a least privilege dbms access model. In *Proceedings of the Fifth Cybersecurity Symposium* (pp. 1-6). <https://doi.org/10.1145/3212687.3212863>
- Ulybyshev, D., Rogers, M., Kholodilo, V., & Northern, B. (2023). End-to-end database software security. *Software*, 2, 163–176. <https://doi.org/10.3390/software2020007>
- Wu, H., Yu, Z., Huang, D., Zhang, H., & Han, W. (2020). Automated enforcement of the Principle of Least Privilege over data source access. In *Proceedings of the IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (pp. 510–517). <https://ieeexplore.ieee.org/document/9343087>
- Yonathan, M. (2025). Access-Controlled Semantic Search: Implementing Role-Based Filtering in Vector Databases for Enterprise Document Management. *Available at SSRN 5360983*. <https://doi.org/10.21203/rs.3.rs-7338381/v1>