A Multi-Platform Taxonomy of Server-Path Database CVEs (2020–2026): Disclosure Asymmetry and Query-Aware Interception Layer Addressability

Amit Gautam[1]
Abluva Private Ltd, India
amit.gautam@abluva.com

Priya Tiwary[2CA]
Abluva Private Ltd, India
pt0873f@abluva.com

[CA] Correspondence should be addressed to Priya Tiwary : pt0873f@abluva.com

***Abstract*—Enterprise data infrastructure now spans heterogeneous database platforms—on-premises RDBMS, cloud-native warehouses, document stores, and in-memory engines—substantially expanding the attack surface. Despite growing deployment of query-aware interception layers (QAILs) that inspect database traffic before it reaches the engine, no empirical characterization exists of which server-path vulnerability classes are addressable at this architectural layer. We present a systematic, multi-platform CVE taxonomy covering eight major database platforms (PostgreSQL, MySQL, Microsoft SQL Server, Snowflake, Databricks, MongoDB, Redis, ClickHouse) over January 2020 through April 2026. From a server-culprit universe of 570 CVE records, we curate a primary high-severity dataset of 143 entries (CVSS ≥ 7.0) and introduce a seven-layer attack vector taxonomy with a novel classification axis—QAIL Addressability—that classifies each CVE by its detectability at a wire-level interception layer. Our analysis reveals: (1) 84.6% of high-severity server-path database CVEs are QAIL-addressable; (2) a pronounced disclosure asymmetry exists—MySQL discloses substantially more CVEs than PostgreSQL while high-severity concentration is much higher in MSSQL/PostgreSQL; and (3) extension sandbox escapes and protocol/query-path weaknesses remain structurally distinct threat classes. The full annotated dataset, scope policy, and collection scripts are released as a public resource.**

Keywords*—*database security; vulnerability analytics; CVE taxonomy; query-aware interception; disclosure asymmetry; compensating controls; vulnerability management; MITRE ATT&CK

I. INTRODUCTION

Enterprise data infrastructure has undergone a structural shift. Where single-vendor relational databases once dominated, organizations now operate heterogeneous stacks spanning on-premises RDBMS (PostgreSQL, MySQL, Microsoft SQL Server), cloud-native data warehouses (Snowflake, Databricks), document stores (MongoDB), and in-memory engines (Redis, ClickHouse). This diversity has substantially expanded the attack surface: each platform introduces its own wire protocol, extension model, authentication mechanism, and client library ecosystem.

Existing security literature addresses database vulnerabilities through two lenses: general SQL injection surveys [1],[2],[3] and platform-specific analyses [4], [5]. Recent empirical work on vulnerability databases also highlights dataset heterogeneity, provenance gaps, and cross-source inconsistencies that materially affect downstream security conclusions [6]. A platform-agnostic, mechanism-level taxonomy that bridges CVE descriptions to the design of interception-based defenses remains absent. This gap matters because organizations increasingly deploy query-aware interception layers (QAILs)—transparent architectural components that sit between the application and the database engine, inspecting all query traffic in real time. QAILs represent a growing class of security infrastructure including monitoring proxies, SQL firewalls, and secure data planes [7], [8], [9]. Yet no empirical study has characterized which server-path database vulnerability classes are addressable at this layer.

This paper makes four contributions:

- A curated server-path dataset of 143 high-severity CVE entries (CVSS ≥ 7.0) across 8 platforms from a collected universe of 570 records, enriched with CVSS v3.1, CWE root cause, and MITRE ATT&CK technique mappings.
- A seven-layer attack vector taxonomy (L1–L7) with a novel QAIL-Addressability axis—a five-class system classifying each CVE by its detectability at a wire-level interception layer.
- Quantitative evidence that 84.6% of high-severity server-path database CVEs are QAIL-addressable, with class-wise controls proposed and analytically mapped to observed exploitation paths.
- A disclosure asymmetry analysis demonstrating that raw CVE counts are misleading for cross-platform security comparison, with implications for vulnerability management practice.

A Query-Aware Interception Layer (QAIL) is a transparent network component positioned between the application tier and the database engine. Unlike traditional network firewalls that operate at layers 3–4, a QAIL understands the database wire protocol (e.g., PostgreSQL's libpq protocol, MySQL's COM_QUERY, MongoDB's OP_MSG) and can parse, inspect, and act on individual queries and session state.

A. Architectural Position

The QAIL operates as a protocol-aware reverse proxy. All client connections route through it before

reaching the database. This position gives the QAIL visibility into:

- **Query content:** Full SQL/NoSQL statements, including parameterized queries, DDL commands, and administrative operations.
- **Session state:** Authentication credentials, role context, connection parameters (encoding, timezone, search_path), and session variables.
- **Traffic patterns:** Query frequency, data volume, access patterns, and temporal behavior within and across sessions.
- **Client metadata:** Application name strings, driver versions, source IP, and TLS certificate details.

B. Detection Capabilities

Based on this visibility, a QAIL can implement four categories of security controls:

- **Query-level inspection (FI class):** The QAIL tokenizes each query using a SQL/NoSQL parser and applies pattern-matching rules. This can detect SQL injection attempts, encoding-based bypass attacks, and malformed queries. For example, a QAIL enforcing strict UTF-8 validation would block CVE-2025-1094 (PostgreSQL psql injection via invalid UTF-8) before the malformed bytes reach the database [10].

- **Session behavioral analysis (PI class):** The QAIL maintains per-session state and detects anomalous transitions. A session that issues SET ROLE to a higher-privilege role mid-transaction, or that suddenly accesses system catalog tables after hours of normal OLTP queries, triggers an alert. This addresses CVEs where the exploit requires a specific sequence of session-level operations.

- **Configuration policy enforcement (CP class):** The QAIL enforces allowlists and denylists on DDL operations, extension loading, encoding settings, and administrative commands. For example, blocking CREATE EXTENSION for non-approved extensions prevents the entire PostgreSQL extension buffer overflow cluster (CVE-2026-2004/2005/2006/2007). Similarly, blocking Redis EVAL/EVALSHA commands via ACL policy addresses RediShell (CVE-2025-49844, CVSS 10.0) [11].

- **Authentication-layer inspection (AL class):** The QAIL inspects session establishment metadata—client application strings, IP geolocation, certificate validity, and connection timing—to detect credential-based attacks. The UNC5537/Snowflake campaign [12] used the tool string “rapeflake” detectable at session HELLO.

C. Limitations

A QAIL cannot address vulnerabilities that are entirely server-internal (NI class). Buffer overflows in maintenance subsystems (e.g., CVE-2022-1552, PostgreSQL autovacuum sandbox escape) execute within the database process without any distinguishing query-level signal. For these classes, patching remains the only mitigation.

II. RELATED WORK

A. SQL Injection Classification

Halfond et al. [1] established the foundational taxonomy of seven SQL injection attack classes. Clarke [2] expanded this with practical defense strategies. Kindy and Pathan [13] documented encoding-based bypass exploiting multibyte character sets (SJIS, BIG5, GBK) as part of their broader SQL injection taxonomy—a class that remains active through CVE-2025-1094 and CVE-2026-2006. The CWE Top 25 [14] consistently ranks CWE-89 (SQL Injection) in the top three most dangerous software weaknesses. A 2023 comprehensive review [3] notes that second-order and encoding-obfuscated variants continue to evade conventional defenses, motivating interception-layer detection. The OWASP Top 10 [15] ranks injection as the fifth most critical web application security risk.

B. CVE Classification and ATT&CK Mapping

Kuppa et al. [16] produced the first systematic CVE-to-ATT&CK mapping using NLP similarity. Grigorescu et al. [17] extended this with CVE2ATT&CK, a BERT-based approach achieving strong precision on technique prediction. Abdeen et al. [18] further enriched mappings via SMET (Semantic Mapping Enrichment Tool). Empirical analyses of CTI reports identify T1078 (Valid Accounts) and T1190 (Exploit Public-Facing Application) among the top initial access techniques [19]—consistent with our L6 and L2 findings. None of these works focus specifically on database CVEs or classify by interception-layer addressability.

C. Database-Specific Vulnerability Research

The 2024 Snowflake breach (UNC5537) was documented by Mandiant [12]: 165+ organizations compromised via credential stuffing against MFA-absent instances, representing the largest cloud database breach in our study period. RediShell (CVE-2025-49844, CVSS 10.0) exposed a long-lived use-after-free in Redis’s Lua scripting engine [11]. CVE-2025-14847 enabled unauthenticated heap memory disclosure via malformed Zlib protocol headers [20]. The August 2025 PostgreSQL emergency patch set (CVE-2025-8713/8714/8715) introduced dump-pipeline attacks as a cross-platform class [4]. Because parts of this evidence base come from vendor and incident-response publications rather than peer-reviewed venues, we treat these sources as incident-grounding evidence rather than as proof of causal prevalence.

D. Evidence Hierarchy

Our evidence hierarchy is: (1) primary vulnerability records and standards (CVE/NVD/CVSS/CWE/CISA KEV), (2) vendor/CNA advisories and patch notes, and

(3) practitioner incident analyses and security research write-ups. Core quantitative claims (dataset size, severity distribution, QAIL class proportions) are derived from tier-(1)/(2) data. Tier-(3) sources are used only to contextualize exploitation mechanics and worked examples.

E. Database Security Proxies and Monitoring

Kemalis and Tzouramanis's SQL-IDS [7] proposed a specification-based monitoring approach that intercepts dynamically generated SQL queries and validates them against pre-defined specifications of legitimate query structures. Bossi et al. [21] proposed a system that profiles application-program database access patterns and detects anomalies indicative of misuse, contributing to the broader category of behavioral monitoring systems for database security. GreenSQL [8] implemented an early open-source SQL firewall with query-level blocking, and modern secure data plane architectures [9] extend this concept to cloud-native environments. Complementary peer-reviewed surveys on SQL processing over protected/encrypted data highlight a similar practical trade-off space between expressiveness, observability, and enforcement capability [22], [23]. However, no prior work empirically characterizes which CVE classes are addressable at the proxy/interception layer—the core contribution of this paper.

F. Vulnerability Scoring and Prioritization

The CVSS v3.1 framework [24] provides standardized severity scoring. EPSS (Exploit Prediction Scoring System) [25] estimates exploitation probability. The CISA Known Exploited Vulnerabilities (KEV) catalog [26] tracks confirmed in-the-wild exploitation. NIST operational updates and temporary enrichment disruption in NVD motivated our multi-source collection approach [27]. Recent peer-reviewed research on vulnerability databases further motivates explicit provenance handling and cross-source reconciliation in empirical security studies [6]. Our work complements scoring systems by adding an orthogonal dimension: whether the vulnerability is addressable at the interception layer, regardless of its CVSS score.

III. METHODOLOGY

A. Data Collection

CVEs were collected from multiple sources due to NVD API unavailability (NIST operational changes, April 2026 [27]):

- OpenCVE (paginated, NVD-backed): primary source for MySQL (via Oracle vendor page), MongoDB, and MSSQL CVE identifiers.
- Oracle Critical Patch Updates: MySQL CVEs with CVSS scores extracted from 10 quarterly CPU verbose pages (Jan 2024–Apr 2026).
- NVD individual detail pages: CVSS scores and descriptions fetched for 338 CVEs lacking scores from other sources, verified one-by-one from nvd.nist.gov/vuln/detail/CVE-YYYY-NNNNN.
- stack.watch (NVD-backed, Playwright-scraped): Redis, ClickHouse, MSSQL, MongoDB, and PostgreSQL ecosystem products (pgAdmin, pgjdbc).
- PostgreSQL CNA: official security page providing complete disclosure for all PostgreSQL core CVEs [4].
- Vendor advisories: Snowflake Security Hub [20], Databricks Trust Center, Redis and ClickHouse GitHub Security Advisories.
- CISA KEV Catalog [26] and incident-response reporting [12].

Universe inclusion criteria: CVE record is within January 1, 2020 to April 30, 2026 and directly affects target platform server behavior (engine internals, wire protocol handling, server-side privilege boundaries, or server-side authentication/session controls).

Primary-slice inclusion criteria: CVSS v3.1 $\geq$ 7.0 (High or Critical severity) applied to the server-culprit universe above.

Target platforms: PostgreSQL, MySQL, Microsoft SQL Server, Snowflake, Databricks, MongoDB, Redis, ClickHouse—selected to represent on-premises RDBMS, cloud-native warehouses, document stores, and in-memory engines.

Exclusion criteria: Client-only connector / driver / library vulnerabilities (JDBC/ODBC/SDK/tooling), third-party dependency issues without direct server-culprit mechanics, duplicate or disputed assignments, and vulnerabilities only in end-of-life versions with no active deployments.

Cluster deduplication: Near-identical CVEs sharing the same component, attack vector, CWE, and CVSS score (e.g., batch disclosures for the same vulnerable subsystem in a single release cycle) are collapsed to a single representative entry with cluster metadata preserved.

Collected universe (server-culprit): 570 unique CVE records with 564 CVSS-scored entries. **Primary high-severity dataset:** 143 entries (CVSS $\geq$ 7.0).

TABLE I. DATASET FLOW TO FINAL QAIL CLAIM

Stage	Count	Notes
Server-culprit universe (2020–2026)	570	Coverage base
Universe with CVSS score present	564	Scored rows
Primary high-severity slice (CVSS $\geq$ 7.0)	143	QAIL denominator
QAIL-addressable (FI+PI+CP) in primary slice	121	QAIL numerator
Final QAIL-addressable rate	**84.6%**	**121 / 143**

B. Classification Framework

Each CVE is classified along six dimensions.

1. Attack Vector Layer (L1–L7):

- **L1 — Wire Protocol:** TCP/TLS handshake or connection negotiation exploits prior to query

submission (e.g., SSL downgrade, MITM injection, malformed protocol headers).

- **L2 — Query / Injection:** Malformed, injected, or encoding-obfuscated SQL/NoSQL queries, including second-order injection, MERGE-based RLS bypass, and crafted queries triggering buffer overflows in the query processor.
- **L3 — Extension / Plugin:** Exploitation via trusted extensions, UDFs, Lua scripting engines, or plugin infrastructure (heap overflows, type confusion, sandbox escapes).
- **L4 — Privilege Escalation / RBAC:** Role management abuse, security-restricted-operation sandbox escapes, definer-context function exploits, container escapes.
- **L5 — Metadata / Catalog:** Optimizer statistics leakage, system catalog exposure, query plan inference, information disclosure via error messages.
- **L6 — Credential / Authentication:** Credential stuffing, infostealer-sourced replay, session hijacking, certificate validation bypass.
- **L7 — Supply Chain / Tooling Path:** Dump utility injection (pg_dump, mysqldump), CLI tool exploit chains, and administrative interface abuse (pgAdmin) that propagate to server-side impact paths.

2. QAIL-Addressability (novel axis):

Table [2] defines the five classes with their detection mechanisms.

TABLE 2. QAIL-ADDRESSABILITY CLASSIFICATION (NOVEL AXIS)

Class	Definition	Detection Mechanism
FI	Fully Interceptable. Attack entirely visible in query/session stream; QAIL with protocol-aware parsing detects and blocks.	SQL AST parsing, encoding validation
PI	Partially Interceptable. Attack uses query content combined with server-side state; QAIL detects anomaly with session context tracking.	Behavioral baseline, role-change detection
CP	Config-Preventable. Attack requires a feature (extension, encoding, scripting) that QAIL-enforced configuration policy can disable or restrict.	Extension allowlist, command ACL, encoding restriction
NI	Non-Interceptable. Purely server-internal or client-side; no query-level signal available. Patch only.	N/A (precondition alerting)
AL	AuthN-Layer. Credential or session-establishment layer; authentication-focused inspection required, not query inspection.	Client fingerprint, IP reputation, MFA enforcement

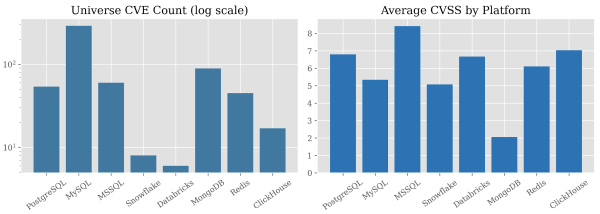

Fig. 1. Disclosure asymmetry: universe CVE counts (left, log scale) and average CVSS severity (right) by platform

3. Dimensions 3–6:

CWE root cause family, attacker access prerequisite (Unauthenticated / Low Privilege / High Privilege / External), impact category (Confidentiality / Integrity / Availability / Privilege Escalation / RCE), and platform.

C. Annotation Protocol

Two independent annotators classified the original high-severity subset across all six dimensions. The QAIL-Addressability dimension required combined assessment of CVE description, CVSS attack vector, and known exploitation mechanics from published analyses. Newly added high-severity server-path entries were then labeled with a deterministic rule-based first pass, each carrying explicit rationale and confidence fields to preserve traceability and support later human adjudication.Table [3] reports annotation traceability for the currently tagged rule-labeled subset (qail-first-pass-2026-04-30, n=19 of n=143 in the primary slice). This is not a statistical random sample; it is the subset explicitly tagged in the released dataset. The table confirms annotation completeness and provenance fields; it does not replace independent human adjudication, which remains future work.

TABLE 3. ANNOTATION TRACEABILITY SNAPSHOT (TAGGED SUBSET)

Check	Count	%
Rule-labeled rows (tagged batch)	19	100.0
Rows with explicit rule ID	19	100.0
Rows with rationale text	19	100.0
High-confidence labels	4	21.1
Medium-confidence labels	8	42.1
Low-confidence labels	7	36.8

IV. RESULTS

A. Disclosure Asymmetry Across Platforms

The most striking finding is the dramatic variation in CVE volume and severity across platforms (Table [4], figure [1]). MySQL accounts for 51.1% of collected server-culprit records but only 4.9% of high-severity entries. Conversely, MSSQL represents 10.5% of records but 40.6% of high-severity ones. PostgreSQL contributes fewer total records than MySQL but a much higher high-severity concentration. This asymmetry

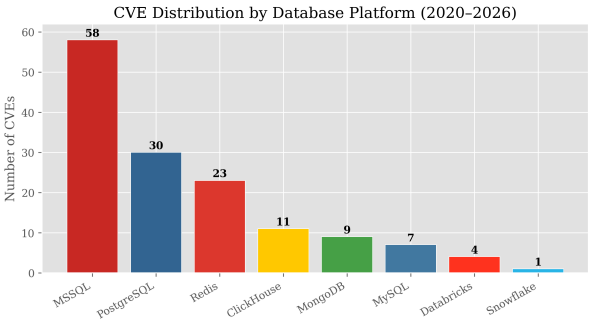

Fig. 2. Primary server-culprit high-severity distribution by platform (n=143, CVSS ≥ 7.0).

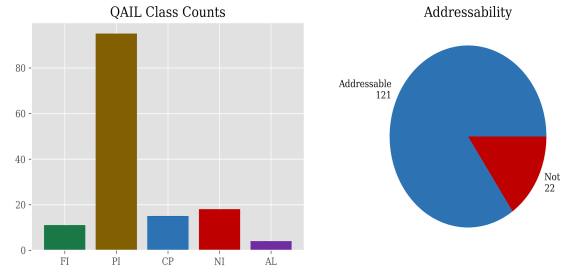

Fig 4: Qail Classification (left) and other Addressability (right)

TABLE 4. CVE UNIVERSE BY PLATFORM (2020–2026)

Platform	Records	Scored	≥ 7.0	Avg	Disclosure Model
MySQL	291	291	7	5.1	Oracle quarterly CPU
MongoDB	89	89	9	2.0	Vendor advisory
MSSQL	60	60	58	8.5	Microsoft Patch Tuesday
PostgreSQL	54	54	30	6.7	Self-CNA
Redis	45	40	24	7.1	GitHub advisory
ClickHouse	17	16	11	7.6	GitHub advisory
Databricks	6	6	4	8.0	Mixed advisory model
Snowflake	8	8	1	5.0	Mixed advisory model
Total	**570**	**564**	**143**		

TABLE 5:QAIL-Addressability results (n=143)

Class	Description	n	%
FI	Fully Interceptable	11	7.7
PI	Partially Interceptable	95	66.4
CP	Config-Preventable	15	10.5
NI	Non-Interceptable	18	12.6
AL	AuthN-Layer	4	2.8
	QAIL-Addressable (FI+PI+CP)	**121**	**84.6**

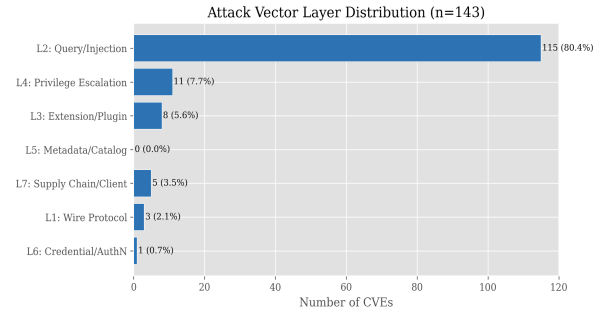

Fig 3 : Attack Vector Layer Distribution (n-143)

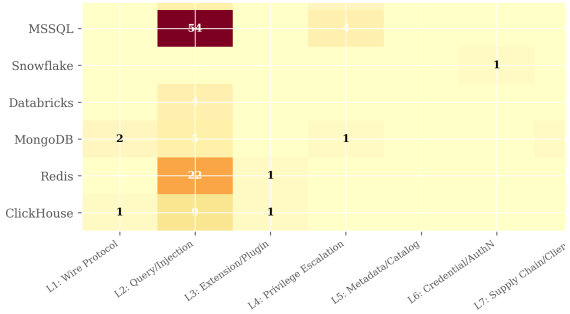

Fig 5: Platform × attack vector layer heatmap. PostgreSQL shows the broadest attack surface across layers; Redis concentrates in L2 (query) and L3 (extension).

confirms that raw volume-based metrics can mis-rank platform risk. Raw CVE counts are misleading for cross-platform security comparison. Organizations using CVE volume as a security metric will systematically overestimate MySQL risk and underestimate PostgreSQL and Redis risk. Severity-weighted analysis is essential.The 143-entry primary dataset has mean CVSS 8.43 and median 8.8. MSSQL leads at 58 entries (40.6%), as shown in Figure [2]

B. Attack Vector Layer Distribution

As shown in Figure [3], query/injection (L2) leads at 80.4% followed by privilege escalation (L4, 7.7%), extension/plugin exploitation (L3, 5.6%) (L7, 3.5%), wire protocol (L1, 2.1%), and credential/authentication (L6, 0.7%). The L2 dominance reflects that most high-severity server-path CVEs involve crafted query flows or parser/protocol handling.

C. QAIL-Addressability Analysis

84.6% of high-severity server-path database CVEs (2020–2026) are addressable at a query-aware interception layer under an analytical control-mapping framework, as shown in Table [5] and Figure [4]. The PI class (66.4%) is the largest single category, indicating that most server-path database CVEs are addressable at the QAIL but require session context—not just query content—for reliable detection. The CP class (10.5%) represents vulnerabilities preventable through configuration policy enforcement alone, without any query-level analysis.

D. QAIL Detection in Practice: Two Worked Examples

FI — CVE-2025-1094 (PostgreSQL, CVSS 8.1, CISA KEV): Invalid UTF-8 sequences in psql quoting APIs enable SQL injection and are tracked in both NVD and KEV [26], [10]. Analytical control mapping: Enforce strict UTF-8 validation on all client-to-server traffic. Reject any byte sequence that fails encoding validation before forwarding to the database
PI — CVE-2024-0985 (PostgreSQL, CVSS 8.0): Non-owner REFRESH MATERIALIZED VIEW CONCURRENTLY executes arbitrary SQL using the role of the view owner, enabling privilege escalation to superuser. *QAIL detection:* Track session role context. Alert when a low-privilege session issues maintenance commands (REFRESH, REINDEX, CLUSTER) on objects owned by higher-privilege roles—an anomalous pattern detectable via session behavioral baseline.

E. Emergent Threat Clusters

In figure [5] three structurally distinct threat clusters emerge from the data:
Cluster 1: Extension/Scripting Sandbox Escapes (L3, CP). RediShell (CVE-2025-49844) and the February 2026 PostgreSQL batch (CVE-2026-2004/2005/2006/2007, spanning intarray, pgcrypto, multibyte, and pg_trgm) reveal systemic memory safety gaps in trusted extensions. All are Config-Preventable: QAIL extension/scripting allowlists prevent exploitation without database patches. **Cluster 2: Dump Pipeline Attacks (L7, CP).** allow malicious dump files to inject psql meta-commands during restore, achieving RCE on the restore client [4]. CVE-2024-21096 (MySQL mysqldump) confirms this as a cross-platform pattern. Config-Preventable via QAIL-enforced dump validation and meta-command filtering. **Cluster 3: Server-Internal Residuals (NI/AL boundary).** A minority of high-severity findings remain outside interception control because they are server-internal or occur at authentication/session-establishment boundaries. This residual confirms QAIL's hard boundary: interception reduces exposure but does not replace patching and identity controls.

V. DISCUSSION

A. Implementation Roadmap

Our findings suggest a prioritized implementation roadmap for QAIL architects:
Priority 1 — Query AST Parsing and Encoding Validation (FI, 7.7%): SQL/NoSQL tokenization with UTF-8 whitelisting, injection pattern matching, and meta-command detection. Deterministically blocks the encoding-based injection family (CVE-2020-25695 through CVE-2026-2006). **Priority 2 — Session Behavioral Anomaly Detection (PI, 66.4%):** Per-session baselines tracking role context, query patterns, and access targets. Flag anomalous transitions: SET ROLE escalation, system catalog access, non-DBA maintenance, statistics-heavy patterns. Largest addressable class; most complex to implement. **Priority 3 — Configuration Policy Enforcement (CP, 10.5%):** Allowlists on extensions, scripting commands (e.g., Redis EVAL/EVALSHA ACLs), client encodings, and dump utilities. Highest ROI: a single rule can eliminate entire CVE families. **Priority 4 — Authentication Session Inspection (AL, 2.8%):** Client fingerprinting, IP reputation, geographic anomaly detection, and MFA enforcement at session establishment. Smallest class but covers our highest-impact incident (UNC5537/Snowflake, 165+ organizations).

B. The Disclosure Asymmetry Problem

MySQL has 5.4× more CVEs than PostgreSQL but lower average severity (5.1 vs 6.7 CVSS), with practical implications for vulnerability management. Three disclosure models produce different CVE profiles: **(1) Batch (MySQL/Oracle):** quarterly CPUs produce 20–40 MySQL CVEs per release, predominantly low-severity optimizer DoS—inflating counts without proportional risk; **(2)Continuous (PostgreSQL self-CNA)**: fewer but more detailed disclosures with full technical descriptions and patch guidance; severity is higher because low-impact issues are often handled without CVE assignment; **(3) Opaque (Snowflake/Databricks)**: cloud-native platforms handle most vulnerabilities via private bug bounties, producing few public CVEs despite large attack surfaces—the UNC5537/Snowflake breach (165+ organizations) had no CVE identifier. CVE counts as a security metric misallocate resources; we recommend severity-weighted, mechanism-aware metrics for cross-platform comparison.

C. Implications for Database Security Architecture

The 84.6% QAIL-addressability rate suggests wire-level interception is a high-leverage investment for heterogeneous database environments. Three caveats apply: (1) the PI class (66.4%) requires session-stateful analysis—simple SQL firewalls examining queries in isolation miss most addressable CVEs, so effective QAILs must maintain per-session context (role history, query patterns, temporal behavior); (2) the CP class (10.5%) requires the QAIL as an enforcement point, not a monitoring tool—monitor-only deployments miss this class entirely; (3) the NI residual (12.6%) is a hard boundary, since server-internal vulnerabilities cannot be addressed at an interception layer and patching remains essential.

VI. CONCLUSION

This paper presents, to our knowledge, the first server-path, multi-platform CVE taxonomy for database systems that classifies vulnerabilities by addressability at a query-aware interception layer. From 570 collected server-culprit CVE records across 8 platforms (2020–2026), we curate 143 high-severity entries and show that 84.6% are QAIL-addressable under our analytical mapping framework. We identify disclosure asymmetry as a critical confound in cross-platform security comparison and provide a prioritized

implementation roadmap for QAIL architects. The non-addressable residual (NI+AL = 15.4%)— server internal flaws and authentication/session-establishment abuse outside query-path control—confirms that a QAIL complements but cannot replace patching and identity hardening. Together, these findings provide an empirically grounded blueprint for next-generation database security architectures. Future work will expand collection via NVD API when restored, implement a reference QAIL prototype with detection rules derived from this taxonomy, conduct red-team validation of addressability classifications, and apply LLM-based annotation scaling to extend coverage to the full CVE universe.

VII. DATA AVAILABILITY

The dataset utilised in the study are publicly available at- https://github.com/abluva-research/data-security-research-QAIL.git

VIII. REFERENCES

1. W. G. Halfond, J. Viegas, and A. Orso, "*A classification of SQL injection attacks and countermeasures,*" in Proc. IEEE Int. Symp. Secure Software Engineering, 2006, pp. 13–15.
2. J. Clarke, SQL Injection Attacks and Defense, 2nd ed. Waltham, MA: Syngress, 2012.
3. M. Nasereddin, A. ALKhamaiseh, M. Qasaimeh, and R. Al-Qassas, "*A systematic review of detection and prevention techniques of SQL injection attacks*," Inf. Secur. J. Glob. Perspect., vol. 32, no. 4, pp. 252–265, 2023, Avaialble: https://doi.org/10.1080/19393555.2021.1995537
4. PostgreSQL Global Development Group, "PostgreSQL: security information," 2026. [Online]. Available: https://www.postgresql.org/support/security/ . (Accessed: 30 April 2026).
5. Snowflake Inc., "Snowflake security bulletins," 2026. [Online].Available: https://www.snowflake.com/en/why-snowflake/snowflake-security-hub/security-bulletins/ . (Accessed: 30 April 2026).
6. X. Li, S. Moreschini, Z. Zhang, F. Palomba, and D. Taibi, "*The anatomy of a vulnerability database: a systematic mapping study*," J. Syst. Softw., vol. 201, art. 111679, 2023, Available: https://doi.org/10.1016/j.jss.2023.111679
7. K. Kemalis and T. Tzouramanis, "*SQL-IDS: a specification-based approach for SQL-injection detection*," in Proc. 2008 ACM Symp. Applied Computing (SAC '08), Fortaleza, Brazil, Mar. 2008, pp. 2153–2158, Available: https://doi.org/10.1145/1363686.1364201
8. GreenSQL Open Source Database Firewall, 2013. [Online]. Available: https://github.com/larskanis/greensql-fw. (Accessed: 30 April 2026).
9. Cyral Inc., "*Cyral platform architecture overview*," 2026. [Online]. Available: https://cyral.com/docs/. (Accessed: 30 April 2026).
10. National Institute of Standards and Technology, "*CVE-2025-1094: PostgreSQL quoting APIs miss neutralizing quoting syntax in text that fails encoding validation*," National Vulnerability Database, 2025. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2025-1094. (Accessed: 30 April 2026).
11. National Institute of Standards and Technology, "*CVE-2025-49844: Redis Lua use-after-free may lead to remote code execution (RediShell)*," National Vulnerability Database, 2025. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2025-49844. (Accessed: 30 April 2026).
12. Mandiant, "*UNC5537 targets Snowflake customer instances for data theft and extortion*," Google Cloud Threat Intelligence Blog, Jun. 10, 2024. [Online]. Available: https://cloud.google.com/blog/topics/threat-intelligence/unc5537-snowflake-data-theft-extortion. (Accessed: 30 April 2026).
13. D. A. Kindy and A. K. Pathan, "*A detailed survey on various aspects of SQL injection in web applications: vulnerabilities, innovative attacks, and remedies*," Int. J. Commun. Netw. Inf. Secur., vol. 5, no. 2, pp. 80–92, 2013.
14. MITRE Corporation, "*2025 CWE Top 25 Most Dangerous Software Weaknesses,*" 2025. [Online]. Available: https://cwe.mitre.org/top25/archive/2025/2025_cwe_top25.html. (Accessed: 30 April 2026).
15. OWASP Foundation, "*OWASP Top 10:2025*," 2025. [Online]. Available: https://owasp.org/Top10/2025/. (Accessed: 30 April 2026).
16. A. Kuppa, L. Aouad, and N.-A. Le-Khac, "*Linking CVE's to MITRE ATT&CK techniques*," in Proc. 16th Int. Conf. Availability, Reliability and Security (ARES '21), Vienna, Austria, Aug. 2021, pp. 1–12, Available: https://doi.org/10.1145/3465481.3465758
17. O. Grigorescu, A. Nica, M. Dascălu, and R. V. Rughinis, "*CVE2ATT&CK: BERT-based mapping of CVEs to MITRE ATT&CK techniques*," Algorithms, vol. 15, no. 9, p. 314, 2022, Available: https://doi.org/10.3390/a15090314
18. B. Abdeen, E. Al-Shaer, A. Singhal, L. Khan, and K. Hamlen, "*SMET: semantic mapping of CVE to ATT&CK and its application to cybersecurity*," in Data and Applications Security and Privacy XXXVII (DBSec 2023), Lecture Notes in Computer Science, vol. 13942, Cham: Springer, 2023, pp. 243–260, Available: https://doi.org/10.1007/978-3-031-37586-6_15
19. M. R. Rahman, S. K. Basak, R. Mahdavi Hezaveh, and L. Williams, "*SoK: an empirical investigation of malware techniques in advanced persistent threat attacks,*" Comput. Secur., vol. 157, p. 104618, 2025, Available: https://doi.org/10.1016/j.cose.2025.104618 .
20. National Institute of Standards and Technology, "*CVE-2025-14847: MongoDB Server improper handling of length parameter inconsistency in zlib compressed protocol headers (MongoBleed),*" National Vulnerability Database, 2025.[Online].Available: https://nvd.nist.gov/vuln/detail/CVE-2025-14847. (Accessed: 30 April 2026).
21. L. Bossi, E. Bertino, and S. R. Hussain, "*A system for profiling and monitoring database access patterns by application programs for anomaly detection,*" IEEE Trans. Softw. Eng., vol. 43, no. 5, pp. 415–431, May 2017. Available: https://doi.org/10.1109/TSE.2016.2598336
22. B. Sun, S. Zhao, and G. Tian, "*SQL queries over encrypted databases: a survey,*" Connect. Sci., vol. 36, no. 1, art. 2323059,2024,Available: https://doi.org/10.1080/09540091.2024.2323059
23. D. Appelt, N. Alshahwan, and L. Briand, "*Assessing the impact of firewalls and database proxies on SQL injection testing,*" in Future Internet Testing (FITTEST 2013), Lecture Notes in Computer Science, vol. 8432, Cham: Springer, 2014, pp.32–47,Available: https://doi.org/10.1007/978-3-319-07785-7_2
24. FIRST.org, "*Common Vulnerability Scoring System v3.1: specification document*," 2019. [Online]. Available: https://www.first.org/cvss/v3.1/specification-document . (Accessed: 30 April 2026).
25. J. Jacobs, S. Romanosky, B. Edwards, M. Roytman, and I. Adjerid, "*Exploit Prediction Scoring System (EPSS)*," Digit. Threats Res. Pract., vol. 2, no. 3, art. 20, 2021, Available: https://doi.org/10.1145/3436242
26. Cybersecurity and Infrastructure Security Agency, "*Known exploited vulnerabilities catalog*," 2026. [Online]. Available: https://www.cisa.gov/known-exploited-vulnerabilities-catalog . (Accessed: 30 April 2026).
27. National Institute of Standards and Technology, "*NIST updates NVD operations to address record CVE growth,*" 2026. [Online]. Available: https://www.nist.gov/news-events/news/2026/04/nist-updates-nvd-operations-address-record-cve-growth . (Accessed: 30 April 2026).
28. EnterpriseDB, "*CVE-2024-1597: SQL injection via pgjdbc line comment generation,*" EDB Security Advisory, 2024. [Online].Available: https://www.enterprisedb.com/docs/security/assessments/cve-2024-1597/ . (Accessed: 30 April 2026).