

Legacy Database Protection Under Assumed Gateway Compromise

Technical white paper

Published by: ViewLegacy LLC.

Author: Asahi Masuda

Publication date: October 2, 2026

1. Executive Summary (1/2)

Assume full attacker control of the cloud gateway: its process, host, runtime memory, environment variables and application logic. API-key authorization, source-IP restrictions, rate limits and gateway-generated audit evidence cease to be reliable security boundaries. Hardening still reduces the probability of compromise; this paper evaluates its consequences when hardening fails.

The evaluated separation keeps DB credentials, raw SQL and executable capability definitions on an uncompromised on-premise agent. The gateway forwards **capability name + parameters**. The agent resolves only locally defined operations, validates inputs, executes prepared SQL and constrains outputs. Database privileges supply an independent boundary. Resource bounds complement this separation [14].

Primary conclusion: removing backend secrets and general execution authority from the gateway reduces immediate credential theft and arbitrary backend operation risks. Agent-side controls can survive gateway failure. The design reduces blast radius; it does not make gateway compromise harmless.

1. Executive Summary (2/2)

A compromised gateway can maliciously call every capability the agent accepts, bypass client scopes, manipulate valid parameters and inspect plaintext business traffic. It can deny service, suppress logs and forge responses delivered to clients. A per-call row cap does not bound cumulative disclosure across repeated calls.

Among the **six specified architecture patterns**, the capability gateway with narrow agent-side contracts and a least-privilege, read-only DB account is the strongest fit for the evaluated read-oriented workload and compromise objective. This is a conditional architectural judgment, not an empirical ranking of six deployed products.

Onprest follows this separation. Its direction is well suited to containing gateway compromise, provided actual capabilities remain narrow and local controls work. It is insufficient when the cloud must never see plaintext, independent caller authorization must survive, writes require strong guarantees, or the agent host is untrusted.

The surviving boundary is the agent, DB privileges and capability/result minimization. Transit confidentiality, trustworthy gateway logs and gateway availability are lost.

2. Background (1/2)



Direct backend authority remains on the compromised host

A gateway with a direct database account concentrates credentials, connection strings, query text and backend authority at an Internet-facing component. Fixed SQL templates improve ordinary request handling, but full host control lets an attacker use the same credential outside those templates. Replication or payload caches add a stored-data theft path.

Architectural separation asks what remains enforceable when the intermediary fails. NIST SP 800-207 rejects implicit trust based only on network location or asset ownership [1]. Applying that principle to the gateway makes the backend resource boundary explicit.

2. Background (2/2)

Attacker powers. Read all gateway memory and environment variables; change authorization and routing; forge API requests; bypass gateway rate/IP checks; read cached OpenAPI metadata; send arbitrary protocol messages; inspect traffic; disconnect the agent; suppress logs and deny service. The attacker can retain intercepted data even when the intended application does not persist it.

Excluded automatic powers. No direct access to the on-premise filesystem or DB network, no agent binary/configuration modification, no agent private-key theft and no alteration of DB privileges. Agent/parser vulnerabilities, malicious dependencies or a second host compromise can invalidate these assumptions; they are not proved absent here.

Evaluated claim. Full gateway control does not by itself confer the agent's filesystem secrets or arbitrary DB execution, if local enforcement and network separation hold. Outbound WebSocket avoids an inbound deployment requirement; it still carries attacker-controlled commands over an established connection.

The reference is primarily read-oriented. Transport security protects a connection from third parties, not plaintext from a compromised processing endpoint [9].

3. Compared Gateway Architectures (1/3)

Pattern	Gateway secrets / settings	SQL knowledge	Business data persisted in cloud
1. Direct DB + arbitrary SQL	DB account / DSN	Full SQL	Not required; logs/caches may add data
2. Direct DB + fixed templates	DB account / DSN	Template text	Not required; logs/caches may add data
3. Broad backend service credential	Service token / integration secrets	Not necessarily	Not required
4. Replication / business cache	Sync / storage secrets; varies	Replication logic may be known	Yes: copied records
5. Stateless generic connector	Connection credentials; varies	May omit SQL until supplied	No intended payload persistence
6. Stateless capability + policy agent	Public agent key; API-key hashes	No raw SQL	No intended payload persistence

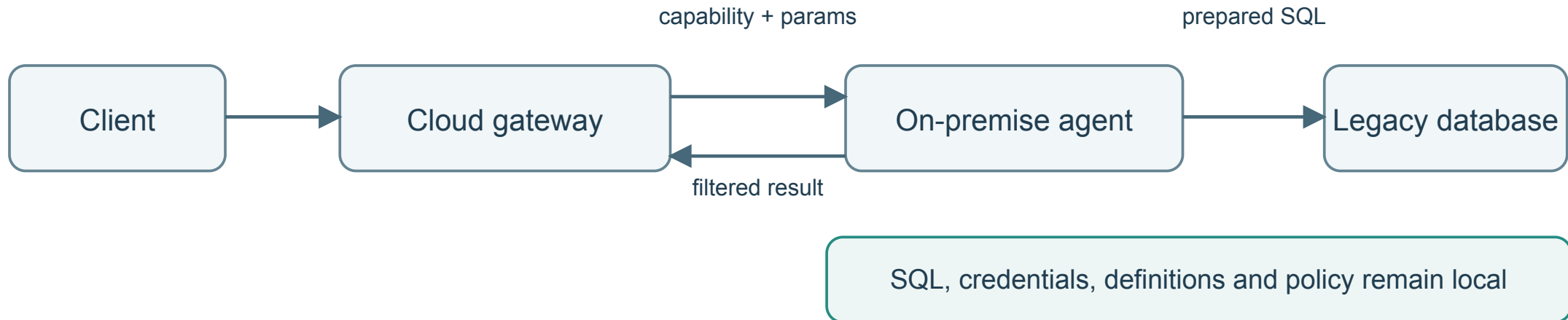
“Stateless” means no intended business-payload persistence; operational logs and volatile metadata still exist. Pattern 6 can expose API tokens in live request memory, even if configured keys are stored as hashes. Backup, APM, debug capture and operator-added caching must be assessed separately.

3. Compared Gateway Architectures (2/3)

Pattern	New arbitrary operations	Physical schema enumeration	Existing operation abuse	Main compromise consequence / residual risk
1. Direct SQL	Within DB privileges	Usually via catalog access	Yes	Credential theft; broad read/write authority
2. Fixed SQL	Yes, outside templates using stolen account	Within account visibility	Yes	Templates on compromised host are bypassable
3. Broad service token	Within backend service API	API-dependent	Yes	Service impersonation; backend ACLs still matter
4. Replication/cache	Sync/backend rights vary	Copied schema visible	Via connected services	Historical data theft; export of cache contents
5. Generic connector	If connector accepts SQL/commands	Often via generic queries	Yes	No cloud secrets, but broad command authority
6. Capability agent	Undefined operations rejected locally	No direct catalog path; metadata inference	Yes, every accepted capability	Narrower execution; transit theft, repetition, DoS

These are patterns, not universal implementations. A backend service with independently enforced narrow permissions can approach pattern 6. A generic connector may be strengthened with local policy; its effective authority then determines the result. No numerical security score is assigned.

3. Compared Gateway Architectures (3/3)



Agent initiates the outbound WebSocket connection

The agent owns `capability.yaml`, DB credentials, SQL, input contracts, execution policy and result allow-lists. It initiates the WebSocket connection. REST, MCP and filtered OpenAPI are gateway-facing interfaces; none grants independent protection if the gateway is fully controlled [10,11].

Prepared parameters preserve query structure. Raw DB errors must be removed **before leaving the agent**. A gateway that merely promises not to log received details cannot protect them after compromise.

4. Security Properties to Evaluate (1/3)

Property after full gateway compromise	Status	Remaining control / qualification
DB credential confidentiality	Preserved under assumptions	Agent placement; DB/network isolation
Raw SQL confidentiality	Preserved under assumptions	Agent-only definitions; local-only detailed errors
Physical schema confidentiality	Partially preserved	No arbitrary catalog query; names/types/errors permit inference
Business-data-at-rest exposure	Partially preserved	No intended cloud copy; DB data still obtainable via capabilities
Business-data-in-transit confidentiality	Lost against gateway	Agent output minimization narrows, but cannot hide plaintext
SQL / schema knowledge inferred from API	Partially preserved	Public OpenAPI and observed results reveal interface semantics

“Preserved” concerns the named asset and stated model, not the entire database. Removing a stored cloud copy reduces historical theft exposure; it cannot stop an attacker from collecting new responses or extracting all rows reachable through accepted operations.

4. Security Properties to Evaluate (2/3)

Attacker objective	Status	Enforcement outside gateway
Execute arbitrary SQL / undefined operation	Preserved if agent is correct	Local operation lookup; no remote SQL execution path
Invoke defined capabilities	Lost as caller-scope guarantee	Agent accepts configured operations; gateway scope bypassed
Manipulate parameters	Partially preserved	Valid values usable; invalid names/types/ranges rejected
Exceed returned row limit	Preserved per call	Agent caps actual rows, independent of input limit
Exceed returned byte limit	Preserved per result	Agent checks filtered serialized result; no oversized payload sent
Perform writes	Conditional	Read-only agent policy plus least-privilege DB account
Bypass agent output filter	Preserved at agent egress	Local allow-list; allowed fields may still contain secrets

A row/byte cap bounds release per call, not DB scan cost, peak intermediate allocation or cumulative data extraction. A `SELECT` may call a function with side effects; keyword classification is not a substitute for restricting function execution and effective DB privileges [7,8].

4. Security Properties to Evaluate (3/3)

Property / objective	Status	Reason
Gateway service availability	Lost	Attacker can disconnect, drop or refuse traffic
Agent signing-key confidentiality	Preserved under assumptions	Private key remains on-premise; public key cannot sign
Forge valid agent signature	Preserved cryptographic distinction	Requires private key under Ed25519 assumptions
Replace agent accepted by gateway	Lost	Hostile verifier can skip checks and connection exclusivity
Gateway audit-log trustworthiness	Lost	Attacker controls emitter, memory, files and forwarding
Client-visible response integrity	Lost	Gateway can alter or fabricate responses after filtering
Agent-local evidence	Partially preserved	Independent host; does not prove actual caller identity or delivery

Ed25519 authenticates the connecting agent to an honest verifier [6]. It neither authorizes each gateway-to-agent business request nor constrains a compromised verifier. Timestamp/nonce/challenge replay defenses depend on the gateway executing its verification rules. Private-key protection and trustworthy gateway admission are different properties.

5. Evaluation Method (1/3)

The reproducible local harness is `src/evaluate.py`. A minimal gateway model normally checks a limited API-key scope, then forwards named requests to an agent model. The compromised path deliberately bypasses that check and gives the attacker the model's gateway state, responses and log controls.

The agent executes parameter-bound SQLite queries over **12 synthetic customers**. Internal secrets are present in DB rows but excluded from the result contract. Public test fixtures contain no real credentials or customer records. Six modeled operations include normal reads, a broad read, an oversized result, a hidden summary and a deliberately slow fixture.

Baseline policy: numeric input `1..8`, actual result cap **3 rows**, filtered-result cap **512 UTF-8 bytes**. Repetition is tested both without an agent quota and with a fixed burst of **3 admissions in 10 calls**. The burst model does not simulate refill, fairness, reconnect or concurrent requests.

Twenty-six scenarios check expected observable outcomes, including successful malicious actions. They do not estimate exploitation probability or the prevention rate.

5. Evaluation Method (2/3)

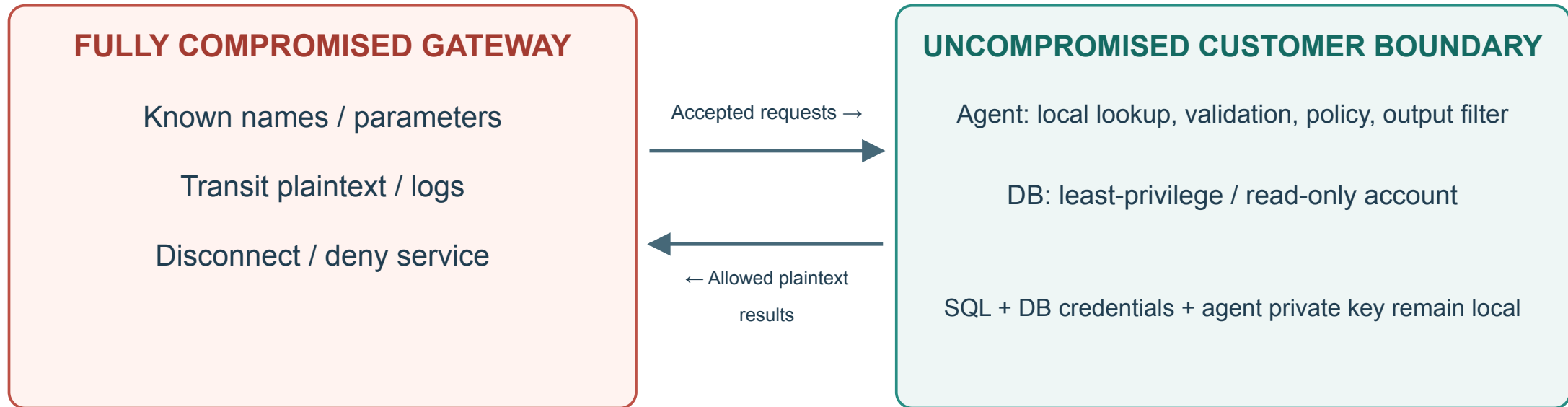
Real library behavior: SQLite parameter binding, read execution, column filtering/serialization in the harness, a SQLite authorizer that rejects writes, and Ed25519 verification with a different private key. The authorizer is a DB-privilege model, not a production role or a test of PostgreSQL/MySQL/SQL Server/Oracle permissions.

Deliberate abstractions: gateway takeover is modeled, not exploited. All objects share one process; host/network isolation is an assumption. A SQLite progress callback aborts a costly recursive read by instruction budget, not a five-second wall-clock deadline. Connection replacement sets a hostile verifier's decision; it is not a forged signature or a live WebSocket takeover.

Alignment to current architecture: `architecture.md` is authoritative [12]. It specifies agent-side optional capability rate limits, bounded concurrency and v2 handshake signatures binding a domain, path, timestamp, nonce, challenge and WebSocket key. Detailed DB errors remain local. The simpler request prompt's three-field signature and error `detail` example are not used as production wire claims.

The harness is an illustrative architectural evaluation, not a formal proof, penetration test or complete security assessment. Current production behavior is read for alignment, not revalidated by this experiment.

5. Evaluation Method (3/3)



The red side is fully controlled. The green boundary survives only while the local agent, definitions and DB privileges remain intact. Accepted capability results still cross back into attacker-visible memory. Protocol parsing must reject or safely handle arbitrary frames without exposing a general executor.

6. Evaluation Results (1/12)

26/26 expected outcomes matched. This means the model behaved as specified, including harmful operations that succeeded. It does not mean 26 attacks were prevented. A01 is a valid-call baseline; A14 observes legitimate traffic; A20/A21 deliberately alter or bypass a local guard to separate agent and DB effects.

Observable result	Local finding	Security interpretation
Undefined operation / arbitrary SQL field	A02/A19 rejected	No remote path to invent operations in this model
Input / result constraints	A03–A09 enforced	Invalid inputs rejected; excess rows limited; bytes rejected; extra columns removed
Repetition without agent quota	A13: 10/10 calls accepted	Gateway rate limiting supplies no surviving control
Repetition with local quota	A23: 3 accepted, 7 rejected	Agent admission survives, but does not restore service availability
Signature vs. connection acceptance	A17 rejected; A24 accepted by hostile verifier	No signature forgery required to corrupt gateway admission
Defined reads, transit, scopes, logs, DoS, result forging	A01/A14–A16/A18/A25/A26 possible	Residual harm remains structurally reachable

Detailed machine-readable observations and assumptions are in `src/evaluation-results.json` [13].

6. Evaluation Results (2/12)

Gateway exposure	Expected after full compromise	Evidence / limitation
DB password / connection string	Absent from intended gateway state	A11 models placement; not host isolation proof
Raw SQL / executable definitions	Absent from intended gateway state	A10 exposes public metadata only
Agent private key	Absent if generation custody is correct	A17 tests wrong-key signature rejection
Agent public key	Readable; attacker can replace verifier config	Public value; no signing authority
Configured API-key hashes / scope mappings	Readable and mutable	Offline hash attack possible; auth logic bypassable
Incoming bearer API keys	Readable while processed	Transient token capture not modeled as a socket test
Business request / response plaintext	Readable while processed	A14; attacker may create persistent stolen copy
Metadata, endpoint identity, logs, infrastructure secrets	Readable if present	Assess TLS termination, telemetry, env and operator tooling

Placement must include provisioning: the gateway CLI can generate an agent keypair, so private-key generation should occur in a trusted environment and leave no gateway copy. Generate or rotate keys on the customer side when feasible. Post-incident rotation is precautionary; DB credentials are not automatically exposed by this model.

6. Evaluation Results (3/12)

Attack-attempt outcomes, A01–A13.

ID	Attempt	Outcome	Observed consequence
A01	Valid existing read	Possible	3 filtered rows returned
A02	Undefined operation	Blocked by agent	Local name resolution rejects
A03	Unknown parameter	Blocked by agent	Input contract rejects extra field
A04	Numeric range > allowed	Blocked by agent	999 rejected
A05	Excess result rows	Limited by agent	Broad SQL yields only 3 rows externally
A06	Excess output bytes	Blocked by agent	2,000-character field exceeds 512 bytes
A07	Write switch on readonly read	Blocked by agent	Undeclared operation parameter rejected
A08	SQL injection through string	No injection	Bound literal yields zero matches
A09	Unlisted DB column	Filtered by agent	internal_secret removed
A10	SQL discovery	Not directly exposed	Metadata contains no raw SQL
A11	Credential discovery	Not directly exposed	No password in gateway view
A12	Physical schema discovery	Partial inference only	Catalog operation absent; API fields visible
A13	Repeated valid calls	Possible	10/10 accepted with no local quota

6. Evaluation Results (4/12)

Attack-attempt outcomes, A14–A26.

ID	Attempt	Outcome	Observed consequence
A14	Intercept legitimate result	Possible	3 plaintext rows observed
A15	Suppress gateway logs	Possible	Gateway log erased; 1 agent event remains
A16	Disconnect / refuse traffic	Possible	Gateway reports offline
A17	Agent signature with wrong key	Blocked by verification	Real Ed25519 verification rejects
A18	Call all known operations	Possible	All 6 known operations attempted without scope checks
A19	Arbitrary SQL in protocol	Blocked by agent	No executable SQL field
A20	Bad local write under readonly	Blocked by agent	Local guard rejects UPDATE definition
A21	Bypass local write guard	Blocked by DB privilege model	SQLite authorizer independently rejects UPDATE
A22	Expensive local read	Blocked by budget model	SQLite progress callback interrupts
A23	Repeat with local burst quota	Limited by agent	3 accepted; 7 rejected
A24	Replace gateway admission decision	Possible	Hostile verifier accepts invalid identity
A25	Known hidden capability	Possible	Hidden summary returns total=12
A26	Forge client-visible response	Possible	Gateway replaces legitimate result

6. Evaluation Results (5/12)

Attack matrix: execution and output.

Attack from compromised gateway	Can attempt?	Agent blocks?	DB blocks?	Business data exposed?	Residual impact
Undefined capability	Yes	Yes	Not reached	No new result	Existing names remain callable
Arbitrary SQL	Yes	Yes: no executor	Not reached	No new result	Protocol/agent flaws outside model
SQL injection parameter	Yes	Binds as data	No extra guard tested	Only literal-match result	Unsafe dynamic local SQL invalidates claim
Excess row request	Yes	Range rejects; actual rows capped	Not guaranteed	Up to allowed cap	Repeat calls; costly scans
Excess byte response	Yes	Rejects oversized result	No	No oversized payload	Intermediate memory/DB cost
Write attempt	Yes	Readonly local policy	Privileges independently limit	No successful write in model	Functions/DB configuration matter
Output field override	Yes	Unknown input rejected; fields filtered	No	Allow-listed fields only	Secrets inside allowed values remain

"No new result" describes the individual blocked attempt; it does not imply that the gateway cannot see other legitimate traffic. A21 isolates the DB privilege model by intentionally bypassing the agent guard.

6. Evaluation Results (6/12)

Attack matrix: accepted authority and gateway control.

Attack from compromised gateway	Can attempt?	Agent blocks?	DB blocks?	Business data exposed?	Residual impact
Valid capability abuse	Yes	No if valid	Only DB privilege boundary	Yes, allowed results	Agent lacks independent client authorization
Repeated calls	Yes	Only if local quota configured	Not generally	Repeated allowed results	Cumulative extraction; workload starvation
Response interception	Yes	No; may minimize	No	Yes, plaintext	Exfiltration from compromised endpoint
Schema discovery	Yes	Unknown operation blocked	Catalog rights vary	API fields / observations	Physical schema inference remains
Credential discovery	Yes	No secret-returning operation	Not applicable	No credential by intended path	Provisioning/debug copies invalidate placement
Gateway log tampering	Yes	No	No	Not inherently	Unreliable gateway evidence
Denial of service	Yes	Cannot restore gateway	Cannot restore gateway	Not required	Gateway refuses or disconnects
Result forgery / agent replacement	Yes	Cannot constrain verifier	No	Not required	Client receives attacker-controlled statements

6. Evaluation Results (7/12)

GATEWAY CONTROLS: LOST

API-key scopes / IP restrictions

Gateway rate limits

Gateway audit integrity / service availability

Transit confidentiality / client-visible integrity

AGENT / DB CONTROLS: CAN SURVIVE

Defined operations / parameter validation

Prepared binding / output allow-list

Row / byte / execution budgets

DB privileges / configured local quotas

Defense in depth still needs both sides. Gateway controls limit ordinary hostile clients. Agent and DB controls limit consequences when those controls are bypassed. Optional local rate limits and bounded concurrency restrict admitted DB work; they cannot prevent the gateway from consuming the quota or cutting the connection.

6. Evaluation Results (8/12)

Trust-boundary table: local authority.

Asset	Stored where?	Visible to gateway?	After compromise?	Primary control
DB credentials	Agent config/runtime	No by design	Not directly	Local custody; least privilege
Raw SQL	Agent config/runtime	No by design	Not directly	Agent-only definitions
Capability names	Agent; published metadata/cache	Published / observed subset	Readable; hidden names may be guessed	Names are not secrets or authorization
Capability definitions	Full definition on agent	Public projection only	Projection readable	Local lookup/policy; reviewed configuration
Agent private key	Agent config/runtime	No by design	Not directly	Trusted key generation and custody
Agent detailed logs	Local agent files	No by design	Not directly	Local-only diagnostics; host access controls
Business data at rest	Legacy DB; customer backups	Via allowed operations only	Reachable subset through calls	DB privileges; narrow reads; output minimization

"Not directly" excludes agent compromise, insecure provisioning, secret-bearing outputs and raw error leakage. Agent definitions should never embed sensitive facts into public descriptions or examples. A published interface is a projection, not the complete local executable contract.

6. Evaluation Results (9/12)

Trust-boundary table: intermediary-visible assets.

Asset	Stored where?	Visible to gateway?	After compromise?	Primary control
Request parameters	Client/agent; gateway transient memory	Yes	Yes; mutable and recordable	Agent validation and minimization
Response body	Agent/client; gateway transient memory	Yes	Yes; mutable and recordable	Agent egress filtering; encryption needed to hide
OpenAPI schema	Agent projection; gateway volatile cache	Yes; full published set	Yes, regardless of API-key filter	No SQL/secrets; non-sensitive interface metadata
API keys	Client; gateway hashes + live bearer tokens	Hashes and transient presented token	Readable; checks can be bypassed	Gateway checks before compromise only
Gateway logs	Stdout/collector/operator storage	Yes	Emitter can lie, suppress or alter	Independent evidence outside gateway
Gateway connection identity / state	Gateway process/host	Yes	Arbitrarily mutable	Honest verification only; external verification needed

A collector can protect events already received from later deletion if it has an independent append-only boundary. It cannot guarantee truthful events or detect every omission by a compromised emitter. Public OpenAPI schemas are not proof of physical DB schema secrecy.

6. Evaluation Results (10/12)

Surviving-controls table.

Control	Enforcing component	What survives	What it does not establish
Undefined capability rejection	Agent	Only local definitions executable	Caller entitlement to each definition
Parameter validation	Agent	Declared types/names/ranges	Ownership, consent or correct business intent
Prepared statements	Agent / DB driver	Parameter values cannot reshape fixed SQL	Safety of dynamic SQL / dangerous stored functions
Actual row / filtered byte caps	Agent	Bounded result release per call	Cumulative extraction / bounded scan cost
Read-only policy	Agent	Read operation classification	Absence of SELECT side effects
Minimum DB privileges	Database	Independent restriction on reads/writes	Correct roles/functions across all engines
Detailed errors kept local	Agent	Raw DB errors do not reach gateway	No inference from public errors/timing
Time / admission budgets	Agent and DB	Modeled execution cancellation; optional local quota	Guaranteed service or verified driver cancellation

The agent, its startup configuration and its DB account are the remaining trusted computing base. Review SQL, output contracts and effective privileges together. No single layer substitutes for the others [2,3,7].

6. Evaluation Results (11/12)

Lost-controls table.

Gateway control	Outcome after full compromise	Consequence	Needed independent boundary
API-key capability scopes	Lost	All accepted local capabilities reachable	Agent-verifiable caller authorization if required
IP restriction	Lost	Hostile process ignores request origin	Independent network policy helps other paths only
Gateway rate limiting	Lost	Attacker generates internal calls directly	Agent quotas/admission; DB workload limits
Gateway audit integrity	Lost	Events can be forged or suppressed	Independent agent evidence / append-only collection
Transit confidentiality	Lost	Request and response plaintext visible	Encryption terminating outside gateway
Gateway availability	Lost	Requests dropped; agent disconnected	Recovery/failover; local workload independence
Client-visible response integrity	Lost	Results, errors and metadata can be forged	Agent-to-client signed, freshness-bound responses
Gateway admission / replay enforcement	Lost as trusted decision	Verifier can accept impostors	Independent verification beyond compromised host

These are negative findings. Deploying an agent is not an authorization tunnel that preserves external user identity automatically. Passing caller identity as an unsigned gateway-controlled parameter does not repair the loss.

6. Evaluation Results (12/12)

Residual-risk table.

Residual risk	Why it remains	Practical consequence
Every accepted capability can be abused	Gateway is the business-request source	Exposure is the union of locally accepted operation authority
Valid parameters target sensitive rows	Validation is not caller authorization	Cross-account disclosure unless independently constrained
Repeated sub-limit reads	Per-call caps do not bound total disclosure	Gradual export; quota exhaustion even with local rate limits
Allowed field contains a secret	Field allow-list is not semantic classification	Sensitive content or inference still leaks
Availability and scan cost	Gateway can refuse service; caps apply to result	Backend pressure, starvation, outage
Physical schema / timing inference	API fields, behavior and observations are visible	Partial knowledge despite raw SQL placement
Agent vulnerability or compromise	Agent/parser/host holds local authority	Credentials, SQL, logs and capabilities may be exposed
Write exposure / over-broad DB permissions	Defined DML still callable after scope bypass	Larger integrity risk; transactions do not cancel every side effect

Risk reduction is multidimensional: credential custody and operation confinement improve, while gateway-visible plaintext and availability do not. A single “safe/unsafe” or aggregate prevention score would hide this distinction.

7. Discussion (1/3)

Capability breadth determines the compromise ceiling. After gateway-side scope checks are bypassed, the attacker can attempt the union of all capabilities the agent accepts, including a known hidden name. Discovery filtering and narrow API keys help normal operation but cannot narrow this post-compromise ceiling.

Narrow capabilities must restrict rows as well as columns. A “get customer” operation accepting any customer ID can disclose records outside the legitimate caller's ownership. An “export all customers” capability may allow complete business-data extraction without arbitrary SQL or credential theft. Read-only privilege reduces destructive authority, not disclosure authority.

A stronger identity requirement needs caller evidence that the agent can independently validate, with audience, operation/parameters, expiry and replay constraints. A gateway-supplied `tenant_id` or unsigned scope statement is attacker-controlled. Such an independent caller-authorization design is future work, not a current guarantee.

Local rate limits reduce throughput and database admission, but an attacker can spend every available token, starve legitimate work and collect results over time. Separate budgets and workload isolation require their own design and tests.

7. Discussion (2/3)

Authentication is directional. Outbound agent authentication proves possession of an agent private key to the gateway while the gateway is honest. It does not prove that a later business request reflects a legitimate external user's authorization. A compromised gateway already controls the authenticated peer the agent talks to.

Current v2 handshake binding, one-use challenges and timestamp/nonce checks strengthen replay protection for an honest verifier [12]. They do not survive as reliable gateway-side acceptance rules under the stipulated host compromise. A public key cannot produce a valid signature, yet a hostile verifier can accept an invalid one. Replacement can deceive clients without replacing the genuine on-premise process.

TLS protects communication from intermediaries outside its endpoints [9]. When the gateway or its controlled runtime processes plaintext, traffic remains visible there. An operator-owned reverse proxy may change the endpoint boundary, but TLS alone does not make the application blind.

Agent output filtering preserves what leaves the agent, not the truth of what the gateway later tells a client. End-to-end response signatures would also need request binding, replay protection, trusted key distribution and client-side verification.

7. Discussion (3/3)

Independent evidence matters. Agent-local events can show an accepted operation and its outcome without trusting gateway logs. They do not prove the actual external caller or that a response reached the client. Agent-signed events in an independent append-only store can strengthen origin and retention; signatures alone do not prove completeness, a secure clock or a truthful compromised signer.

Availability remains exposed. Timeouts, bounded concurrency, row/byte caps and DB workload settings help contain work. A compromised gateway can still refuse all traffic, disconnect the tunnel or fill the admitted workload. DB confidentiality and integrity can remain partially protected during a service outage.

Additional compromise changes the claim. If the agent host is also compromised, its credentials, SQL, capabilities and logs can become visible. A DB role still supplies an independent privilege ceiling, but stolen credentials may be usable from other permitted network locations.

Keep gateway-side controls in production. They reduce ordinary attack opportunities; local agent and DB boundaries reduce consequences after those controls fail. The architecture should be described as blast-radius reduction, not gateway-independent security.

8. Practical Architecture Guidance (1/3)

1. Keep DB credentials, DSNs, raw SQL and full executable definitions on the customer side. Separate gateway credentials from agent/DB secrets. Avoid cloud payload stores, caches and debug/APM capture; inspect operator-added logging too.
2. Resolve only local capability names. Validate parameters at the agent, use prepared binding, enforce actual row/filtered byte/time limits and reject disallowed output fields. Keep raw database errors local before any gateway transmission.
3. Review row predicates, joins, allowed values and output fields for every capability. Minimize capability breadth, result breadth and the business information crossing the tunnel. Avoid administrative catalogs, general query tools and dangerous functions.
4. Use non-owner, non-superuser DB roles with the least table/view/function privileges needed. Prefer read-only credentials for read services. A read-only transaction mode is supplementary and engine-specific [7,8].
5. Configure agent-side rate limits and bounded admission appropriate to DB capacity. Bound scan work and intermediate allocation separately from returned-result caps. Treat cancellation/driver behavior as a real integration-test obligation.

8. Practical Architecture Guidance (2/3)

1. Document the owner of each control: gateway for normal client authentication/scopes/IP; agent for local definitions/contracts/admission; DB for independent privileges; infrastructure for network separation, TLS termination, host custody and rebuild.
2. Keep gateway logs free of parameters, response bodies, secrets and raw DB errors. After compromise, treat their origin and completeness as suspect. If stronger forensic guarantees are required, retain independent agent evidence outside the gateway [4].
3. Prepare incident isolation, clean gateway rebuild, external credential revocation and reconnection procedures. Rotate API keys, agent keys and DB credentials as appropriate to the assessed exposure; rotation does not undo intercepted plaintext [5].
4. Generate private keys only in trusted custody and remove provisioning copies. Prefer short-lived/narrowly scoped credentials where practical. A signature is useful only with an uncompromised verifier and trusted public-key distribution.
5. Test full-gateway-control scenarios explicitly with real binaries and DB privileges before relying on them. Validate malformed frames, scope bypass, quotas, repeated reads, raw-error separation, cancellation and connection replacement; record negative outcomes too.

This paper provides evidence for the local model only. It does not mark production incident-response or red-team readiness as completed.

8. Practical Architecture Guidance (3/3)

Which design is best for which requirement?

Requirement	Preferred direction	Reason / cost
Narrow read-only integration; gateway compromise containment	Pattern 6 + narrow contracts + least-privilege DB + agent quotas	Removes cloud DB secrets/general executor; maintains bounded local authority
Gateway must not see business plaintext	Add client-to-agent payload encryption / gateway-blind relay	Key custody and discovery/client compatibility become more complex
Caller scopes must survive gateway takeover	Agent-verifiable independent authorization	Need request binding, identity provider trust and replay controls
Tamper-resistant forensic evidence	Independent agent audit + append-only retention	Requires separate credentials, monitoring and completeness analysis
Writes or broad administrative operations	Strong approval/workflow and DB transaction design	Larger integrity blast radius; stronger authorization and reconciliation
Untrusted agent host / high-assurance isolation	Evaluate isolation, attestation, TEE or different architecture	Larger trusted computing base and operational cost; not automatic superiority

Assessment of Onprest: the core separation is the preferred foundation for the first row. The paper does not claim that today's deployment satisfies every stronger row or is universally the best architecture.

9. Limitations and Future Work (1/2)

The evaluation is illustrative, not a substitute for a full audit or penetration test. It models neither all remote-code-execution techniques nor host escape, malicious drivers, parser bugs, side channels or supply-chain attacks. The same-process harness assumes host/network separation rather than establishing it.

Gateway-visible plaintext remains visible; TLS alone does not protect data from a processing endpoint. Existing capabilities remain callable, DoS remains possible, gateway logs remain corruptible and client-visible responses remain forgeable. The integrity of the agent host is critical; its compromise can expose credentials, SQL, definitions and detailed logs.

Database read-only behavior depends on engine privileges, roles and functions. SQLite's authorizer models a privilege boundary; it is not evidence for a production DB role. The instruction-budget timeout and fixed-burst quota do not validate real driver cancellation, a wall-clock deadline, refill, concurrency or reconnect behavior.

The focus is read-oriented. Writes have a larger blast radius and need independent authorization, approval where appropriate, idempotency, transactional controls, affected-row bounds and reconciliation. Incident rebuild, key rotation and recovery procedures are not fully evaluated. No production security certification or cross-product benchmark is implied.

9. Limitations and Future Work (2/2)

Future direction	Potential benefit	Trade-off / verification needed
Application-layer / gateway-blind encryption	Hide payload from gateway runtime	Client/agent key custody; replay/binding; metadata still visible
TLS pass-through	Avoid gateway TLS decryption	Endpoint must move; plaintext REST/MCP inspection/routing may change
Dedicated isolated decryption component	Separate payload authority	Isolation fails if gateway host can control the component
Confidential computing / enclave / TEE	Limit host/operator access under specific model	Attestation, side channels, key release, availability and platform trust
WASM / process isolation	Reduce execution-component authority	Host remains trusted; permissions/resources and escape risk
Agent-signed audit / append-only storage	Strengthen origin and retention	Completeness, clock, independent credentials; protect signer
Remote attestation / end-to-end response signing	Bind measured code / verify response origin	Measurement/key trust, freshness, replay and rollout complexity
Rebuild, key rotation, writes and real red team	Validate operational failure contracts	Real binaries, multiple engines, incident drills and external review

These approaches address different requirements and are not automatically better. Privileged cloud operators, regulatory separation, endpoint compromise and high-assurance isolation require an explicit threat model and verified implementation before choosing them.

10. References (1/3)

[1] [NIST SP 800-207 — Zero Trust Architecture](#)

No implicit trust based solely on network location or ownership.

[2] [OWASP — SQL Injection Prevention Cheat Sheet](#)

Prepared binding separates parameter values from query structure.

[3] [OWASP — Input Validation Cheat Sheet](#)

Server-side allow-list validation; syntax and semantic checks.

[4] [OWASP — Logging Cheat Sheet](#)

Sensitive-data exclusion and protection of logging systems.

External source URLs verified during preparation. References support control mechanisms, not a security certification of this architecture.

10. References (2/3)

[5] [OWASP — Secrets Management Cheat Sheet](#)

Least privilege and secret lifecycle, revocation and rotation.

[6] [IETF RFC 8032 — Edwards-Curve Digital Signature Algorithm \(EdDSA\)](#)

Ed25519 signing and verification; not gateway authorization.

[7] [PostgreSQL 18 — Privileges](#)

SELECT, write and EXECUTE privilege boundaries and ownership.

[8] [PostgreSQL 16 — SET TRANSACTION](#)

Read-only mode has engine-specific semantics and exceptions.

External source URLs verified during preparation. References support control mechanisms, not a security certification of this architecture.

10. References (3/3)

[9] [IETF RFC 8446 — The Transport Layer Security \(TLS\) Protocol Version 1.3](#)

Transport-channel confidentiality and authenticated endpoints.

[10] [OpenAPI Specification v3.1.1](#)

API schema describes an interface; it is not physical DB schema.

[11] [Model Context Protocol — Tools \(2025-11-25\)](#)

Tool metadata and calls do not replace server-side enforcement.

[14] [OWASP API4:2023 — Unrestricted Resource Consumption](#)

Resource and request limits reduce abuse; availability remains an explicit objective.

[12] **Internal architecture record.** `architecture.md`, gateway/agent and policy sections; reviewed for this paper. Source of product alignment, not independent assurance.

[13] **Local evaluation.** `src/evaluate.py` and `src/evaluation-results.json`; 26 illustrative scenarios, 12 synthetic rows. Supporting material in `src.zip`.

External source URLs verified during preparation. References support control mechanisms, not a security certification of this architecture.

In Closing

The strongest fit among the six patterns for narrow read-only integration is a stateless capability gateway with contracts enforced on-premise, minimum DB privileges and local admission controls. It removes DB credentials and raw SQL from the cloud boundary and limits executable operations to reviewed local definitions.

Onprest's core direction fits this conclusion. Its advantage is containment of backend authority. The gateway can still abuse accepted operations, see plaintext, forge client-visible results, suppress its logs and stop service. Stronger confidentiality, identity, audit or write requirements need additional independent controls.

Published by: ViewLegacy LLC.

Author: Asahi Masuda

Publication date: October 2, 2026