

ゲートウェイ侵害を前提としたレガシーデータベースの保護

技術白書

発行: 合同会社ビューレガシー

著者: 増田朝陽

発行日: 2026年10月02日

1. エグゼクティブサマリー (1/2)

クラウドゲートウェイのプロセス、ホスト、実行時メモリ、環境変数、アプリケーションロジックを攻撃者が完全に支配すると仮定する。APIキー認可、送信元IP制限、レート制限、ゲートウェイ生成の監査証拠は、信頼できるセキュリティ境界として扱えなくなる。堅牢化は侵害確率を下げるが、本書はその防御が破られた後の影響を評価する。

評価する分離方式では、DB認証情報、生SQL、実行可能なケイパビリティ定義を、侵害されていないオンプレミスエージェントに置く。ゲートウェイはケイパビリティ名とパラメータを転送する。エージェントがローカル定義だけを解決し、入力検証、プリペアドSQL実行、出力制約を行う。DB権限が独立した境界を補強する。資源上限がこの分離を補完する[14]。

主な結論：ゲートウェイからバックエンド秘密と汎用実行権限を取り除くと、認証情報の直接窃取と任意操作のリスクを縮小できる。エージェント側統制は残りうるが、ゲートウェイ侵害が無害になるわけではない。

1. エグゼクティブサマリー (2/2)

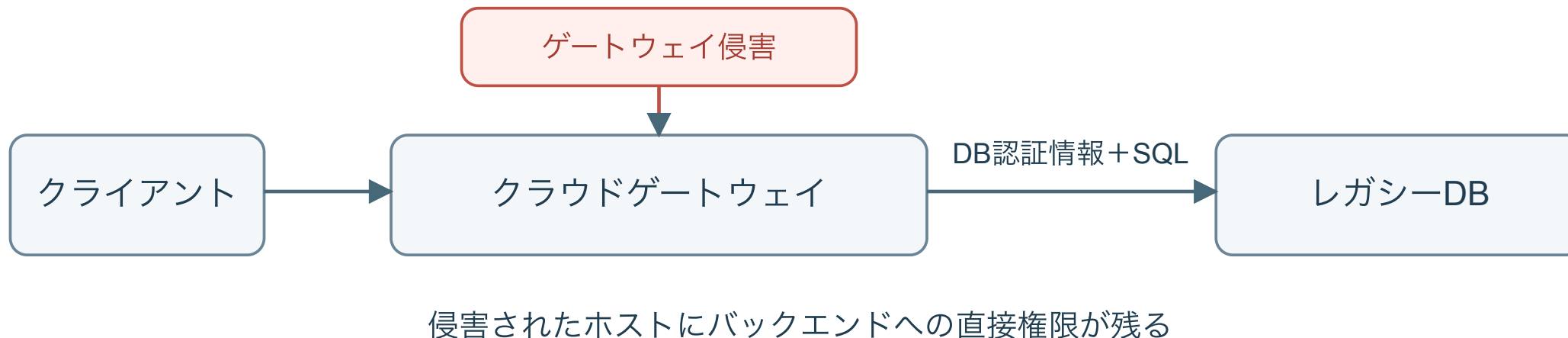
侵害済みゲートウェイは、エージェントが受理する全操作を悪用し、利用者別スコープを迂回し、有効なパラメータを操作し、平文の業務通信を閲覧できる。サービス停止、ログ抑止、利用者に返す応答の偽造も可能である。1回あたりの行上限は、反復呼出しによる累積開示を制限しない。

指定された6方式の中では、狭い契約をエージェントで強制し、最小権限・読取専用DBアカウントを組み合わせたケイパビリティ方式が、評価対象の読取業務と侵害後の被害縮小に最も適する。これは条件付きの設計判断であり、6つの稼働製品の実測順位ではない。

Onprestはこの分離方針を採っており、操作を狭く保ち、ローカル統制が機能するなら方向性は妥当である。ただし、クラウドでの平文閲覧禁止、侵害後も残る利用者認可、強い書込保証、信頼できないエージェントホストへの対応には追加対策が必要になる。

残る境界はエージェント、DB権限、操作・結果の最小化である。通過中の機密性、ゲートウェイログの信頼性、ゲートウェイの可用性は失われる。

2. 背景 (1/2)



DBアカウントを直接持つゲートウェイは、認証情報、接続文字列、クエリ本文、バックエンド権限をインターネット向けコンポーネントへ集中させる。固定SQLテンプレートは通常時の要求を制約するが、ホストを完全支配した攻撃者は、同じ資格情報をテンプレート外で使える。複製やペイロードキャッシュには、保存データの窃取経路も加わる。

設計上の分離は、中継側が破られても何を強制できるかを問う。NIST SP 800-207は、ネットワーク位置や所有だけに基づく暗黙の信頼を認めない[1]。この考え方をゲートウェイにも適用し、バックエンド資源を守る境界を明確にする。

2. 背景 (2/2)

攻撃者に認める能力。ゲートウェイの全メモリ・環境変数の読取、認可・ルーティングの変更、API要求の偽造、ゲートウェイ側rate/IP検査の迂回、OpenAPIキャッシュの読取、任意プロトコルメッセージの送信、通信の閲覧、エージェント切断、ログ抑止、サービス妨害。通常アプリが保存しなくても、攻撃者は傍受データを保存できる。

自動的には認めない能力。オンプレミスのファイルシステムやDBネットワークへの直接到達、エージェントのバイナリ・設定変更、秘密鍵の窃取、DB権限の変更。エージェントやパーサの脆弱性、悪意ある依存、別ホストの侵害で前提が崩れる可能性はあり、本書はその不存在を証明しない。

評価する主張。ローカル強制とネットワーク分離が成立すれば、ゲートウェイの完全支配だけでは、エージェント内の秘密や任意DB実行権限を得られない。outbound WebSocketはinbound公開要件をなくすが、確立済み接続では攻撃者の命令を受けうる。

対象は主に読取業務である。通信暗号化は第三者から接続を守るが、平文を処理する侵害済み終端からは守らない[9]。

3. ゲートウェイ方式の比較 (1/3)

方式	ゲートウェイの秘密・設定	SQL知識	クラウドの業務データ永続化
1. DB直接接続＋任意SQL	DBアカウント／DSN	SQL全体	必須ではない。ログ・キャッシュ次第
2. DB直接接続＋固定SQL	DBアカウント／DSN	テンプレート本文	必須ではない。ログ・キャッシュ次第
3. 広いバックエンド資格情報	サービストークン／連携秘密	必ずしも持たない	必須ではない
4. 複製／業務キャッシュ	同期・保存用秘密。方式次第	複製処理を知りうる	あり：複製レコード
5. ステートレス汎用コネクタ	接続用資格情報。方式次第	送信前はSQLを持たない場合もある	意図的なペイロード永続化なし
6. ケイパビリティ＋ポリシー エージェント	エージェント公開鍵、APIキーhash	生SQLなし	意図的なペイロード永続化なし

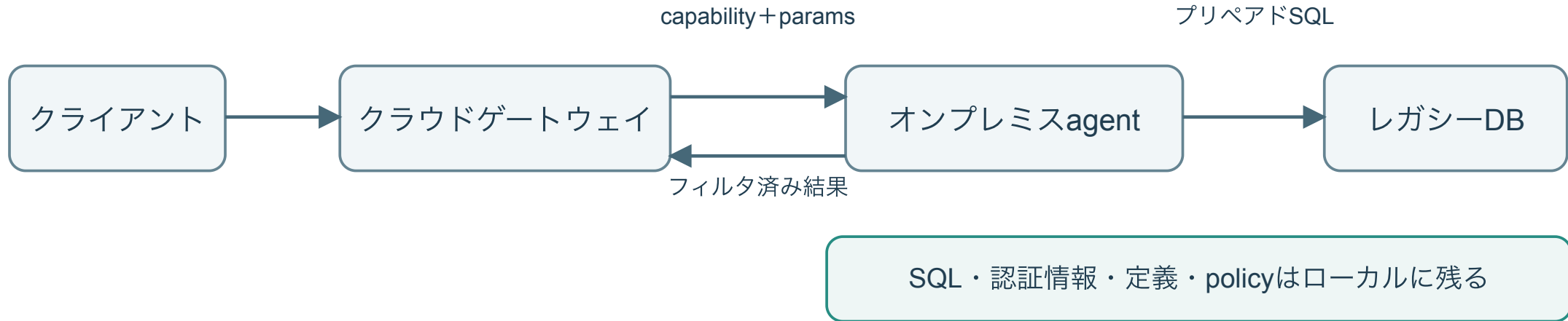
「ステートレス」は業務ペイロードを意図的に永続化しない意味であり、運用ログや揮発メタデータは存在する。方式6でも、設定キーがhash保存されていても、受信APIトークンは実行時メモリで露出する。バックアップ、APM、debug取得、運用者が加えたキャッシュは別途確認する。

3. ゲートウェイ方式の比較 (2/3)

方式	新しい任意操作	物理スキーマの列挙	既存操作の悪用	主な侵害影響・残存リスク
1. 任意SQL	DB権限内で可能	通常はcatalogで可能	可能	資格情報窃取、広い読取・書込
2. 固定SQL	窃取アカウントで枠外実行	アカウントの可視範囲	可能	侵害ホスト上のテンプレートは迂回可能
3. 広いservice token	バックエンドAPIの範囲	API次第	可能	サービスなりすまし。下流ACL次第
4. 複製／キャッシュ	同期・下流権限次第	複製スキーマは見える	接続サービス経由	過去データ窃取、キャッシュ全体流出
5. 汎用コネクタ	SQL・汎用命令を受理するなら可能	汎用クエリなら可能	可能	クラウド秘密なしでも実行権限が広い
6. ケイパビリティ	未定義操作はローカル拒否	直接catalog経路なし。推論は残る	受理する全操作で可能	実行範囲縮小。傍受、反復、DoSは残る

これは方式の比較であり、全実装に共通する断定ではない。下流が狭い権限を独立して強制するサービスなら方式6に近づく。汎用コネクタもローカルポリシーで強化でき、その実効権限が結果を決める。数値による安全性スコアは付けない。

3. ゲートウェイ方式の比較 (3/3)



agentがoutbound WebSocket接続を開始する

エージェントが `capability.yaml`、DB認証情報、SQL、入力契約、実行ポリシー、結果allow-listを所有し、WebSocket接続を開始する。REST、MCP、APIキー別OpenAPIはゲートウェイ側の公開面であり、ゲートウェイを完全支配された場合に独立した防御を与えない[10,11]。

パラメータをbindしてクエリ構造を固定する。生のDBエラー詳細はエージェントを出る前に除去する。受信した詳細をログに残さないというゲートウェイ側の約束だけでは、侵害後に守れない。

4. 評価するセキュリティ特性 (1/3)

完全侵害後の特性	状態	残る統制・条件
DB認証情報の機密性	前提の下で維持	エージェント側配置、DB・ネットワーク分離
生SQLの機密性	前提の下で維持	エージェント限定定義、詳細エラーのローカル保持
物理スキーマの機密性	部分的に維持	任意catalog照会なし。名称・型・エラーから推論可能
保存業務データの露出縮小	部分的に維持	意図的なクラウド複製なし。操作経由の取得は可能
通過中業務データの機密性	ゲートウェイに対して喪失	エージェントの最小化で量は減るが平文は隠せない
APIから推論するSQL・schema知識	部分的に維持	OpenAPIと観測結果がインターフェースの意味を示す

「維持」は対象資産と前提についての表現であり、DB全体の安全を意味しない。クラウド保存コピーの除去は過去データの窃取を減らす
が、新しい応答の収集や、受理操作で到達できる全行の取得を防がない。

4. 評価するセキュリティ特性 (2/3)

攻撃者の目的	状態	ゲートウェイ外の強制
任意SQL・未定義操作の実行	エージェントが正しければ維持	ローカル操作解決。remote SQL実行経路なし
定義済み操作の呼出し	利用者スコープ保証は喪失	設定操作は受理され、ゲートウェイ スコープを迂回
パラメータ操作	部分的に維持	有効値は使える。不正な名前・型・範囲は拒否
返却行上限の超過	1回ごとには維持	入力limitとは独立して実行結果を制限
返却byte上限の超過	結果ごとには維持	フィルタ後の直列化結果を検査し、過大結果を送らない
書込実行	条件付き	エージェント読取専用ポリシーとDB最小権限
エージェント出力フィルタの迂回	エージェント出口では維持	ローカルallow-list。許可列内の秘密は残りうる

行・byte上限は1回の外部出力を制限し、DB走査コスト、中間メモリ最大量、累積取得を制限しない。`SELECT`でも副作用のある関数を呼びうるため、先頭keyword分類は関数実行権限や実効DB権限の制限に代わらない[7,8]。

4. 評価するセキュリティ特性 (3/3)

特性・目的	状態	理由
ゲートウェイサービスの可用性	喪失	切断・破棄・受付拒否が可能
エージェント署名秘密鍵の機密性	前提の下で維持	秘密鍵はオンプレ。公開鍵では署名できない
正しいエージェント署名の偽造拒否	暗号的な区別は維持	Ed25519の前提の下で秘密鍵が必要
ゲートウェイが受理するエージェントの置換	喪失	検証と接続数制限を侵害側が無効化できる
ゲートウェイ監査ログの信頼性	喪失	発行処理・メモリ・ファイル・転送を支配
利用者に届く応答の完全性	喪失	フィルタ後の応答を改変・偽造できる
エージェントローカル証拠	部分的に維持	独立ホスト。ただし本人性や実際の配達は証明しない

Ed25519は、正直な検証者に対して接続エージェントを認証する[6]。業務要求ごとのゲートウェイ→エージェント認可や、侵害済み検証者の制約にはならない。timestamp・nonce・challengeのreplay対策も、ゲートウェイが検証規則を実行する前提を持つ。秘密鍵保護と、信頼できる接続受付は別の特性である。

5. 評価方法 (1/3)

再現可能なローカルハネスは `src/evaluate.py` である。最小ゲートウェイモデルは通常、限定されたAPIキースコープを検査してから、名前付き要求をエージェントモデルへ転送する。侵害経路ではこの検査を意図的に迂回し、ゲートウェイ状態・応答・ログ制御を攻撃者に与える。

エージェントは架空顧客12件に対して、パラメータをbindしたSQLiteクエリを実行する。DB行には内部秘密があるが、結果契約から除外する。公開テストfixtureに実認証情報や実顧客情報は含めない。通常読取、広い読取、過大結果、非公開summary、意図的に遅いfixtureの6操作をモデル化する。

基準ポリシーは、数値入力1～8、実結果3行、フィルタ後結果512 UTF-8 bytes。反復はエージェント 呼出し予算なしと、10回中burst 3回を受理する設定で比較する。burstモデルは補充、公平性、再接続、並行要求を模擬しない。

26シナリオでは、攻撃が成功する場合も含めて観測結果を確認する。侵害確率や防御成功率は推定しない。

5. 評価方法 (2/3)

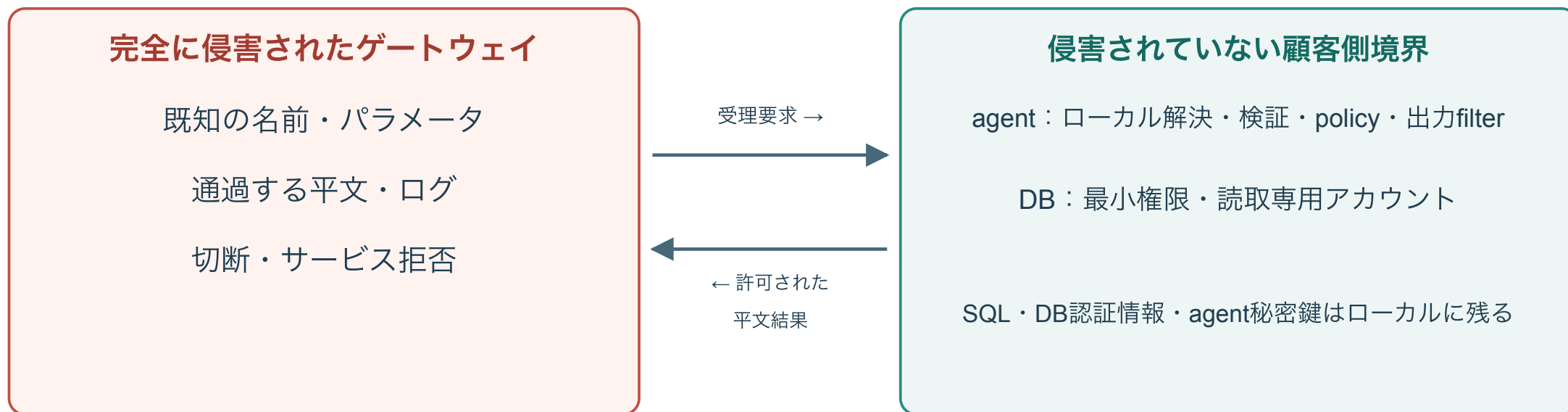
実際に動かすライブラリ処理：SQLiteのパラメータbindと読取、ハーネスの列フィルタ・直列化、書込を拒否するSQLite authorizer、異なる秘密鍵によるEd25519検証。authorizerはDB権限のモデルであり、本番ロールやPostgreSQL／MySQL／SQL Server／Oracleの権限試験ではない。

意図的な抽象化：ゲートウェイ乗っ取りはモデル化し、脆弱性を突かない。全オブジェクトが1プロセスを共有し、ホスト・ネットワーク分離は前提とする。SQLite progress callbackで高コスト再帰読取を命令予算で中断し、5秒の実時間期限は測らない。接続置換は侵害検証者の判断を変更するモデルであり、署名偽造や実WebSocket乗っ取りではない。

現行設計との対応：正本は `architecture.md` である[12]。エージェント側の任意capability rate制限、有限並列枠、domain・path・timestamp・nonce・challenge・WebSocket keyを束縛するv2署名を規定する。DB詳細エラーはローカルに保つ。依頼文の3要素署名とエラー `detail` 例は、本番wire仕様として使わない。

本実験は設計の例示評価であり、形式証明、侵入試験、包括的監査ではない。現行実装は整合確認のために読み、本実験で再検証したとは扱わない。

5. 評価方法 (3/3)



赤い領域は攻撃者が完全に支配する。緑の境界は、ローカルエージェント、定義、DB権限が健全な間だけ残る。受理された操作の結果は、攻撃者に見えるメモリへ戻る。任意frameは、汎用executorを公開せずに拒否または安全に処理しなければならない。

6. 評価結果 (1/12)

26件すべてで想定した観測結果と一致した。これは、有害な操作が成功する場合も含め、モデルが想定どおり動いた意味である。26件の攻撃を防いだ意味ではない。A01は正常呼出し、A14は通常通信の観測、A20・A21はエージェントとDBの効果を分けるためにローカルguardを意図的に変更・迂回する。

観測項目	ローカル結果	セキュリティ上の解釈
未定義操作・任意SQL フィールド	A02・A19を拒否	モデルでは新操作をremote生成する経路なし
入力・結果制約	A03～A09が機能	不正入力拒否、行数制限、byte超過拒否、余分な列除去
エージェント 呼出し予算なしの反復	A13：10回すべて受理	ゲートウェイ rate制限は侵害後に残らない
ローカル呼出し予算ありの反復	A23：3回受理、7回拒否	エージェント 受付制御は残るが、サービス可用性は戻らない
署名と接続受理の違い	A17拒否、A24は侵害検証者が受理	ゲートウェイ受付の改変に署名偽造は不要
既存読取・傍受・スコープ・ログ・DoS・応答偽造	A01・A14～A16・A18・A25・A26で可能	残存被害の経路は依然として存在

機械可読な観測結果と前提は `src/evaluation-results.json` に収録する[13]。

6. 評価結果 (2/12)

ゲートウェイ側の露出対象	完全侵害後の想定	証拠・限界
DB password・接続文字列	意図するゲートウェイ状態にはない	A11は配置モデル。ホスト分離証明ではない
生SQL・実行可能定義	意図するゲートウェイ状態にはない	A10は公開メタデータだけを返す
エージェント秘密鍵	生成時の管理が適切ならない	A17は別鍵署名の拒否を確認
エージェント公開鍵	閲覧でき、検証設定も変更可能	公開値であり署名権限はない
設定APIキーhash・スコープ map	閲覧・変更可能	offline hash攻撃や認証処理迂回が可能
受信Bearer APIキー	処理中に閲覧可能	socket経由の一時token取得は未試験
業務要求・応答の平文	処理中に閲覧可能	A14。攻撃者が窃取コピーを永続化できる
メタデータ・endpoint・ログ・基盤秘密	存在すれば閲覧可能	TLS終端、telemetry、env、運用ツールも確認

配置には鍵配布も含む。ゲートウェイ CLIでエージェント鍵ペアを生成できるため、信頼できる環境で生成し、ゲートウェイに秘密鍵のコピーを残さない。可能なら顧客側で生成・更新する。侵害後の鍵更新は予防策であり、本モデルだけでDB認証情報の漏洩を断定しない。

6. 評価結果 (3/12)

攻撃試行の結果 A01～A13。

ID	試行	結果	観測した影響
A01	定義済みの正常読取	可能	フィルタ済み3行を返却
A02	未定義操作	エージェントが拒否	ローカルname解決で拒否
A03	未知パラメータ	エージェントが拒否	入力契約で余分なフィールド拒否
A04	数値範囲超過	エージェントが拒否	999を拒否
A05	実結果行数の超過	エージェントが制限	広いSQLでも外部返却は3行
A06	出力byte超過	エージェントが拒否	2,000文字フィールドが512 bytes超過
A07	readonly読取でwrite切替	エージェントが拒否	未宣言操作パラメータ拒否
A08	文字列のSQL注入	注入できない	bindしたliteralで一致0件
A09	非許可DB列	エージェントが除去	internal_secretを除去
A10	SQL探索	直接は露出しない	メタデータに生SQLなし
A11	資格情報探索	直接は露出しない	ゲートウェイ視界にpasswordなし
A12	物理schema探索	推論は残る	catalog操作なし。API フィールドは見える
A13	正常操作の反復	可能	ローカル呼出し予算なしで10/10受理

6. 評価結果 (4/12)

攻撃試行の結果 A14～A26。

ID	試行	結果	観測した影響
A14	正常応答の傍受	可能	平文3行を観測
A15	ゲートウェイログ抑止	可能	ゲートウェイログ消去。エージェントに1event残る
A16	切断・受付拒否	可能	ゲートウェイがofflineを返す
A17	別鍵でエージェント署名	検証が拒否	実Ed25519検証で拒否
A18	全既知操作でスコープ迂回	可能	既知6操作をスコープ検査なしで試行
A19	プロトコルへ任意SQL	エージェントが拒否	実行可能なSQL フィールドなし
A20	readonly下の不正ローカル write	エージェントが拒否	ローカル guardがUPDATE定義を拒否
A21	ローカル write guardを迂回	DB権限モデルが拒否	SQLite authorizerが独立してUPDATE拒否
A22	高コストローカル read	予算モデルが拒否	SQLite progress callbackで中断
A23	ローカル burst 呼出し予算下の反復	エージェントが制限	3回受理、7回拒否
A24	ゲートウェイ受付判断の置換	可能	侵害検証者が不正本人性を受理
A25	名前を知る非公開操作	可能	非公開summaryがtotal=12を返す
A26	利用者向け応答の偽造	可能	正規結果をゲートウェイが置換

6. 評価結果 (5/12)

攻撃マトリクス：実行と出力。

侵害ゲートウェイからの攻撃	試行可能？	エージェントが拒否？	DBが拒否？	業務データ露出？	残存影響
未定義capability	可能	する	未到達	新規結果なし	既存名の呼出しは残る
任意SQL	可能	する：executorなし	未到達	新規結果なし	プロトコル・エージェント脆弱性は対象外
パラメータのSQL注入	可能	dataとしてbind	追加guard未試験	literal一致結果だけ	ローカル動的SQLがあれば前提崩壊
過大な行要求	可能	範囲拒否・実行結果制限	保証しない	上限内の行	反復・高コスト走査
過大なbyte応答	可能	過大結果を拒否	しない	過大ペイロードなし	中間メモリ・DBコスト
書込試行	可能	ローカル readonly ポリシー	権限が独立して制限	モデルで書込成功なし	関数・DB設定次第
出力フィールド上書き	可能	未知入力拒否・列除去	しない	許可列だけ	許可値内の秘密は残る

「新規結果なし」は、その拒否された試行に対する表現であり、別の正常通信をゲートウェイが見られない意味ではない。A21はエージェント guardを意図的に迂回し、DB権限モデルの効果を分離する。

6. 評価結果 (6/12)

攻撃マトリクス：既存権限とゲートウェイ支配。

侵害ゲートウェイからの攻撃	試行可能？	エージェントが拒否？	DBが拒否？	業務データ露出？	残存影響
正常capabilityの悪用	可能	有効ならしない	DB権限境界のみ	許可結果を露出	独立した利用者認可がない
反復呼出し	可能	ローカル 呼出し 予算設定時のみ	通常しない	許可結果を反復取得	累積取得、処理枠の占有
応答傍受	可能	しない。最小化は可能	しない	平文を露出	侵害終端から流出
schema探索	可能	未知操作は拒否	catalog権限次第	API フィールド・観測結果	物理schemaの推論は残る
資格情報探索	可能	秘密を返す操作なし	対象外	意図した経路で露出なし	生成・debugコピーで前提崩壊
ゲートウェイログ改ざん	可能	しない	しない	必須ではない	ゲートウェイ証拠が信頼できない
DoS	可能	ゲートウェイを復旧できない	ゲートウェイを復旧できない	不要	受付拒否・切断
結果偽造・エージェント置換	可能	検証者を制約できない	しない	不要	利用者に攻撃者の内容を返す

6. 評価結果 (7/12)

ゲートウェイ統制：喪失

APIキーscope・IP制限

gatewayレート制限

gateway監査の完全性・サービス可用性

通過中の機密性・利用者向け応答の完全性

agent・DB統制：残りうる

定義済み操作・パラメータ検証

プリペアドbind・出力allow-list

行・byte・実行予算

DB権限・設定したlocal quota

多層防御には両側が必要である。ゲートウェイ統制は通常の敵対的クライアントを制限する。エージェント・DB統制は、それが迂回された後の影響を制限する。任意のローカル rate制限と有限並列枠はDB処理への入場を狭めるが、ゲートウェイによる呼出し予算占有や接続切断を防がない。

6. 評価結果 (8/12)

信頼境界：ローカル権限。

資産	保存先	ゲートウェイから見える？	侵害後に見える？	主な保護
DB認証情報	エージェント設定・実行時	設計上は見えない	直接は見えない	ローカル管理、最小権限
生SQL	エージェント設定・実行時	設計上は見えない	直接は見えない	エージェント限定定義
capability名	エージェント、公開メタデータ・cache	公開・観測した一部	閲覧可能。非公開名も推測可能	名称は秘密や認可ではない
capability定義	全定義はエージェント	公開projectionだけ	projectionは見える	ローカル解決・ポリシー・設定レビュー
エージェント秘密鍵	エージェント設定・実行時	設計上は見えない	直接は見えない	信頼できる鍵生成・管理
エージェント詳細ログ	エージェントローカルfile	設計上は見えない	直接は見えない	ローカル診断、ホスト access制御
保存業務データ	legacy DB、顧客バックアップ	許可操作経由だけ	操作で到達できる一部	DB権限、狭い読取、結果最小化

「直接は見えない」は、エージェント侵害、安全でない配布、秘密を返す出力、生エラー漏洩を除外する。公開説明やexampleに機密情報を埋め込まない。公開インターフェースはprojectionであり、ローカルの実行契約全体ではない。

6. 評価結果 (9/12)

信頼境界：中継側から見える資産。

資産	保存先	ゲートウェイから見える？	侵害後に見える？	主な保護
要求パラメータ	クライアント・エージェント、ゲートウェイ時メモリ	見える	閲覧・変更・保存可能	エージェント入力検証・最小化
応答本文	エージェント・クライアント、ゲートウェイ時メモリ	見える	閲覧・変更・保存可能	エージェント出口フィルタ。隠すには暗号化必要
OpenAPI schema	エージェント projection、ゲートウェイ揮発cache	公開集合全体が見える	APIキーフィルタに関係なく見える	SQL・秘密を含めない公開メタデータ
APIキー	クライアント、ゲートウェイ hash・受信token	hashと受信時token	閲覧可能。検査を迂回可能	侵害前のゲートウェイ検査のみ
ゲートウェイログ	stdout・collector・運用保存先	見える	生成側が嘘・抑止・改変可能	ゲートウェイ外の独立証拠
ゲートウェイ接続本人性・state	ゲートウェイ プロセス・ホスト	見える	任意に変更可能	正直な検証。必要なら外部検証

独立した追記専用境界を持つcollectorは、受信済みeventの後日の削除を防ぎうる。しかし、侵害生成側が送る内容の真実性や、全抑止の検出は保証しない。公開OpenAPI schemaだけで物理DB schemaの秘匿を証明できない。

6. 評価結果 (10/12)

残る統制。

統制	強制するコンポーネント	残る効果	証明しないこと
未定義capability拒否	エージェント	ローカル定義だけを実行	各定義への利用者の正当権限
パラメータ検証	エージェント	宣言した型・名前・範囲	所有権、同意、業務意図の正しさ
プリペアドstatement	エージェント・DB driver	値で固定SQL構造を変えられない	動的SQL・危険なstored 関数の安全性
実行結果行・フィルタ後byte上限	エージェント	1回の返却量を限定	累積取得・走査コストの上限
読取専用ポリシー	エージェント	読取操作の分類	SELECT副作用の不存在
DB最小権限	DB	読取・書込の独立した制約	全engineでのロール・関数設定の正しさ
詳細エラーのローカル保持	エージェント	DB生エラーをゲートウェイへ出さない	公開エラー・timingからの推論不存在
時間・受付制御予算	エージェント・DB	モデルの実行中断、任意ローカル 呼出し予算	サービス保証・driver cancelの実証

残る信頼基盤は、エージェント、起動設定、DBアカウントである。SQL、出力契約、実効権限をまとめてレビューする。どの1層も、他の層の代わりにはならない[2,3,7]。

6. 評価結果 (11/12)

失われる統制。

ゲートウェイ統制	完全侵害後	影響	必要な独立境界
APIキーのcapability スコープ	喪失	受理するローカル操作すべてに到達	必要ならエージェントが検証する利用者認可
IP制限	喪失	侵害プロセスが送信元を無視	独立ネットワーク ポリシーは別経路を制約
ゲートウェイ rate制限	喪失	内部呼出しを直接生成	エージェント 呼出し予算・受付制御、DB負荷制限
ゲートウェイ監査の完全性	喪失	eventの偽造・抑止	独立エージェント証拠・追記専用収集
通過中データの機密性	喪失	要求・応答の平文を閲覧	ゲートウェイ外で終端する暗号化
ゲートウェイ可用性	喪失	要求破棄、エージェント切断	復旧・failover、ローカル処理独立
利用者向け応答の完全性	喪失	結果・エラー・メタデータを偽造	エージェント→クライアント署名とfreshness束縛
ゲートウェイ受付・replay強制	信頼できる判断を喪失	検証者が偽装相手を受理	侵害ホスト外の独立検証

これらは否定的な結果である。エージェント設置だけで外部利用者の本人性が自動的に保たれるわけではない。署名のない本人性をゲートウェイ制御のパラメータで渡しても、失われた認可を修復できない。

6. 評価結果 (12/12)

残存リスク。

残存リスク	残る理由	実務上の影響
受理する全capabilityの悪用	ゲートウェイが業務要求の送信元	ローカル受理操作の権限の合計が露出範囲
有効パラメータで機密行を指定	検証は利用者認可ではない	独立制約がなければ他accountを開示
上限内読取の反復	1回の上限は累積開示を制限しない	段階的流出、ローカル rate下でも呼出し予算占有
許可列内に秘密を含む	フィールド allow-listは意味分類ではない	機密内容や推論による漏洩
可用性・走査コスト	ゲートウェイは拒否可能。上限は結果側	下流負荷、処理枠占有、停止
物理schema・timing推論	API フィールド、挙動、観測が見える	生SQL配置を分離しても一部知識を得る
エージェント脆弱性・侵害	エージェント・parser・ホストがローカル権限を持つ	資格情報、SQL、ログ、capabilityが露出
書込公開・過大DB権限	スコープ迂回後も定義DMLを呼べる	完全性リスク拡大。transactionで全副作用は消せない

リスク縮小は多次元である。資格情報の管理と実行範囲は改善するが、ゲートウェイが見る平文と可用性は改善しない。単一の「安全／危険」や合算防御率では、この違いが隠れてしまう。

7. 考察 (1/3)

ケイパビリティの広さが侵害後の上限を決める。ゲートウェイ スコープ検査を迂回すると、既知の非公開名を含め、エージェントが受理する全capabilityを試せる。発見フィルタと狭いAPIキーは通常運用に役立つが、侵害後の権限上限を狭めない。

狭い操作は列だけでなく行も制限する必要がある。任意の顧客IDを受け取る「顧客取得」は、正規利用者の所有範囲外のレコードを返す。 「全顧客export」は、任意SQLや資格情報窃取なしで業務データ全体を取り出せる。読取専用権限は破壊権限を減らし、開示権限は減らさない。

強い本人性が必要なら、エージェントが独立検証できる利用者証拠にaudience、操作・パラメータ、期限、replay制約を束縛する必要がある。ゲートウェイが渡す `tenant_id` や未署名スコープは攻撃者の制御下にある。この独立認可設計は今後の検討であり、現行の保証ではない。

ローカル rate制限は処理量とDB 受付制御を下げるが、攻撃者は全tokenを使い、正規処理を妨げ、長期に結果を集められる。予算分離と負荷隔離には、独自の設計と試験が必要になる。

7. 考察 (2/3)

認証には方向がある。outbound エージェント認証は、ゲートウェイが正直な間、エージェント秘密鍵の所有をゲートウェイへ証明する。その後の業務要求が、正規外部利用者の認可を反映することは証明しない。侵害ゲートウェイは、エージェントが通信している認証済みpeerそのものを支配している。

現行v2のhandshake束縛、1回限りのchallenge、timestamp・nonce検査は、正直な検証者へのreplayを強化する[12]。ただし、想定するホスト侵害後にゲートウェイ受付規則を信頼できるわけではない。公開鍵で正しい署名を作れなくても、侵害検証者は不正署名を受理できる。オンプレの正規プロセスを置き換えなくても、利用者を欺ける。

TLSは終端外の中継者から通信を守る[9]。ゲートウェイやその実行環境が平文を処理すれば、そこで閲覧できる。環境所有のreverse proxyで終端境界は変わるが、TLSだけでアプリが内容を見なくなるわけではない。

エージェント出力フィルタはエージェントから出る内容を制約し、ゲートウェイが後で利用者へ伝える真実性は守らない。end-to-end応答署名にも、要求束縛、replay防止、信頼できる鍵配布、クライアント検証が必要になる。

7. 考察 (3/3)

独立した証拠が重要になる。エージェント ローカル eventは、ゲートウェイログに頼らず、受理した操作と結果を示しうる。ただし、本当の外部利用者やクライアントへの応答到達は証明しない。エージェント署名eventを独立した追記専用storeへ送ると、発行元と保持を強化できるが、署名だけで完全な記録、正しい時計、侵害署名者の真実性は保証しない。

可用性は依然として露出する。timeout、有限並列枠、行・byte上限、DB負荷設定は処理を抑える。一方、侵害ゲートウェイは全通信の拒否、tunnel切断、受理枠の占有を行える。サービス停止中も、DBの機密性・完全性が部分的に守られる場合はある。

追加侵害で主張が変わる。エージェント ホストも侵害されれば、認証情報、SQL、capability、ログが見える。DB ロールには独立した権限上限が残るが、盗まれた資格情報を別の接続許可ネットワークから使われる可能性がある。

本番のゲートウェイ統制は維持する。通常攻撃を減らし、エージェント・DB境界はそれが破られた後の影響を減らす。設計は被害範囲縮小と表現し、ゲートウェイに依存しない安全性とは表現しない。

8. 実践的な設計指針 (1/3)

1. DB認証情報、DSN、生SQL、完全な実行定義を顧客側に置く。ゲートウェイ資格情報とエージェント・DB秘密を分離する。クラウドのペイロード store、cache、debug・APM取得を避け、運用者が追加したログも確認する。
2. ローカル capability名だけを解決する。エージェントで入力検証、prepared bind、実行結果の行・フィルタ後byte・時間上限、非許可出力列の除去を行う。DB生エラーはゲートウェイ送信前にローカルへ閉じる。
3. 各操作の行条件、join、許可値、出力フィールドをレビューする。操作の広さ、結果の広さ、tunnelを通る業務情報を最小化する。管理 catalog、汎用query tool、危険な関数を避ける。
4. 非owner・非superuserのDB ロールへ、必要なテーブル・view・関数権限だけを付ける。読取serviceには読取専用資格情報を優先する。読取専用transaction modeは補助でありengine依存である[7,8]。
5. DB容量に合わせてエージェント rate制限と有限受付制御を設定する。返却量の上限とは別に、走査負荷と中間メモリを制約する。driver cancelの挙動は実結合試験で確認する。

8. 実践的な設計指針 (2/3)

1. 各統制の所有先を明記する。ゲートウェイは通常のクライアント認証・スコープ・IP、エージェントはローカル定義・契約・受付制御、DBは独立権限、基盤はネットワーク分離・TLS終端・ホスト管理・再構築を担う。
2. ゲートウェイログにパラメータ、応答本文、秘密、DB生エラーを残さない。侵害後は発行元と完全な記録を疑う。強い事後調査保証が必要なら、ゲートウェイ外に独立エージェント証拠を保持する[4]。
3. インシデント隔離、cleanなゲートウェイ再構築、外部資格情報の失効、再接続を準備する。露出評価に応じてAPIキー、エージェント鍵、DB資格情報を更新する。更新しても傍受済み平文は取り戻せない[5]。
4. 秘密鍵は信頼できる管理下で生成し、配布用コピーを消す。可能なら短命・狭い資格情報を使う。署名は検証者と公開鍵配布が健全な場合だけ有効である。
5. 実binaryとDB権限で完全ゲートウェイ支配を明示的に試験してから依拠する。不正frame、スコープ迂回、呼出し予算、反復読取、生エラー分離、cancel、接続置換を確認し、否定的結果も記録する。

本書の証拠はローカルモデルだけに対するものであり、本番インシデント対応やred-team準備を完了扱いにしない。

8. 実践的な設計指針 (3/3)

要件ごとに、どの設計が最適か。

要件	適する方向	理由・負担
狭い読取連携、ゲートウェイ侵害後の被害縮小	方式6+狭い契約+DB最小権限+エージェント 呼出し予算	クラウド DB秘密・汎用executorを除去し、ローカル権限を限定
ゲートウェイに業務平文を見せない	クライアント→エージェント ペイロード暗号化・blind relayを追加	鍵管理、発见面、クライアント互換性が複雑になる
ゲートウェイ乗っ取り後も利用者スコープ維持	エージェントが検証する独立認可	要求束縛、本人性 providerの信頼、replay制御が必要
改ざん耐性のある事後調査証拠	独立エージェント監査+追記専用保持	別資格情報、監視、記録の完全性分析が必要
書込・広い管理操作	強い承認workflowとDB transaction設計	完全性への影響が大きく、強い認可と結果照合が必要
信頼できないエージェント ホスト・高保証隔離	隔離・attestation・TEE・別設計を評価	信頼基盤と運用負担が増える。自動的に優れるわけではない

Onprestの評価：基本の分離方式は、最初の要件に対する最適な土台である。現行deploymentが強い要件をすべて満たすことや、あらゆる用途で最良であることは主張しない。

9. 限界と今後の課題 (1/2)

本評価は例示であり、包括的監査や侵入試験の代替ではない。全RCE手法、ホスト escape、悪意あるdriver、parser不具合、side channel、supply chain攻撃は模擬しない。同一プロセスのハーネスでは、ホスト・ネットワーク分離を実証せず前提とする。

ゲートウェイが処理する平文は見えるままであり、TLSだけでは処理終端から保護できない。既存操作を呼べ、DoSを起こせ、ゲートウェイログを改変でき、利用者向け応答も偽造できる。エージェント ホストの健全性が重要で、その侵害では認証情報、SQL、定義、詳細ログが露出する。

DB読取専用の効果はengine権限、ロール、関数に依存する。SQLite authorizerは権限境界のモデルであり、本番DB ロールの証拠ではない。命令予算timeoutと固定burst 呼出し予算は、実driver cancel、実時間期限、補充、並行処理、再接続を検証しない。

主な対象は読取である。書込は影響範囲が広く、独立認可、必要な承認、冪等性、transaction制御、更新行上限、結果照合が要る。インシデント再構築、鍵更新、復旧手順は十分評価していない。本番の安全認証や製品間benchmarkを意味しない。

9. 限界と今後の課題 (2/2)

今後の方向	期待できる効果	負担・必要な検証
application層・ゲートウェイ blind暗号化	ゲートウェイ 実行環境からペイロードを隠す	クライアント・エージェント鍵管理、replay・束縛。メタデータは残る
TLS pass-through	ゲートウェイでTLS復号しない	終端の移動。平文REST・MCP検査やrouting変更
独立した復号コンポーネント	ペイロード権限の分離	ゲートウェイ ホストが制御できるなら隔離不成立
Confidential computing・enclave・TEE	特定モデルでホスト・運用者の閲覧を制限	attestation、side channel、鍵解放、可用性、platform信頼
WASM・プロセス隔離	実行コンポーネントの権限縮小	ホスト信頼は残る。権限・資源・escape検証
エージェント署名監査・追記専用store	発行元・保持を強化	完全な記録、時計、独立資格情報、署名者保護
remote attestation・end-to-end応答署名	測定code束縛・応答発行元確認	測定・鍵信頼、freshness、replay、配布の複雑さ
再構築・鍵更新・書込・実red team	運用失敗時の契約を検証	実binary、多DB、インシデント訓練、外部レビュー

これらは異なる要件を扱い、自動的に優れるわけではない。特権クラウド運用者、規制上の分離、endpoint侵害、高保証隔離には、選択前に明示的な脅威モデルと実装検証が必要になる。

10. 参考文献 (1/3)

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

位置や所有だけに基づく暗黙の信頼を置かない。

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

パラメータbindで値とquery構造を分離する。

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

server側allow-list検証、構文と意味の検査。

[4] [OWASP — Logging Cheat Sheet](#)

機密情報の除外とlog systemの保護。

外部URLは作成時に確認。文献は統制の仕組みを支え、本設計の安全認証を意味しない。

10. 参考文献 (2/3)

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

最小権限、秘密のlifecycle、失効と更新。

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

Ed25519署名と検証。ゲートウェイ認可ではない。

[7] [PostgreSQL 18 — Privileges](#)

SELECT・write・EXECUTE権限とownershipの境界。

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

読取専用modeにはengine固有の意味と例外がある。

外部URLは作成時に確認。文献は統制の仕組みを支え、本設計の安全認証を意味しない。

10. 参考文献 (3/3)

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

transport channelの機密性と認証された終端。

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

API schemaは公開面を表し、物理DB schemaではない。

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

tool メタデータとcallはserver側強制の代替ではない。

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

資源・要求上限で悪用を減らし、可用性を別途評価する。

[12] 内部設計正本。 `architecture.md` のゲートウェイ・エージェント・ポリシー節。本書作成時に確認。製品整合の根拠であり、独立保証ではない。

[13] ローカル評価。 `src/evaluate.py`、 `src/evaluation-results.json`。例示26シナリオ、架空12行。補助資料は `src.zip` に収録。

外部URLは作成時に確認。文献は統制の仕組みを支え、本設計の安全認証を意味しない。

最後に

狭い読取連携では、6方式の中で、オンプレの契約強制・DB最小権限・ローカル 受付制御を組み合わせたステートレスケイパビリティ方式が最も適する。クラウド境界からDB認証情報と生SQLを取り除き、実行操作をレビュー済みのローカル定義に限定する。

Onprestの基本方針はこの結論に合う。強みはバックエンド権限の封じ込めにある。一方、ゲートウェイは受理操作を悪用し、平文を見て、利用者向け結果を偽造し、ログを抑止し、サービスを止められる。より強い機密性、本人性、監査、書込要件には、追加の独立統制が必要になる。

発行: 合同会社ビューレガシー

著者: 増田朝陽

発行日: 2026年10月02日