International Journal of Science and Research Archive
eISSN: 2582-8185
Cross Ref DOI: 10.30574/ijsra
Journal homepage: https://ijsra.net/

(REVIEW ARTICLE)

Check for updates

Security-by-design for large language model platforms in google cloud platform: preventative controls for vertex AI agents with controlled external tool access

Ranjan Kathuria *

Staff Cloud Security Engineer, Information Security of Rubrik, USA.

International Journal of Science and Research Archive, 2026, 19(02), 151-158

Publication history: Received on 27 March 2026; revised on 02 May 2026; accepted on 05 May 2026

Article DOI: https://doi.org/10.30574/ijsra.2026.19.2.0992

Abstract

Large language model platforms are increasingly integrated into enterprise workflows, where internal artificial intelligence agents assist with tasks such as reviewing digital artifacts, summarizing technical content, and analyzing code, tickets, or documentation; code review with GitHub is one representative example of these patterns. While such systems improve productivity, they introduce new risks involving data exfiltration, over-privileged tool use, prompt injection, secret exposure, incomplete logging, and unauthorized automated actions0.

This research addresses the problem of securing an internal platform for large language model-based agents built on Google Cloud Platform using Vertex AI as the model layer and a Model Context Protocol style integration for interacting with external tools such as source control or issue-tracking systems. Following a secure-by-design methodology, the paper proposes a preventative security architecture that applies hard infrastructure, networking, identity, and monitoring boundaries before runtime interactions occur. The proposed design uses VPC Service Controls to place Vertex AI and related Google-managed services inside an API-level service perimeter that reduces data-exfiltration risk, combines Private Service Connect interfaces and egress proxies to keep agent traffic on controlled private paths, applies Identity and Access Management Deny policies to enforce non-bypassable guardrails on sensitive cloud operations, stores all tool credentials in Secret Manager with encryption at rest, and constrains agent behavior through narrowly scoped Model Context Protocol tools that expose only non-destructive actions to external systems.

In addition, the architecture centralizes observability by enabling detailed audit, access, and trace logging for large language model calls, network flows, and tool invocations, exporting this telemetry to a security information and event management platform to support detection, response, and quantitative risk assessment. The design is evaluated using a quantitative risk formula based on likelihood and impact, and the results show that the proposed architecture reduces modeled platform risk by approximately 91.33%, indicating that preventative infrastructure, identity, and monitoring controls can materially improve the security posture of enterprise large language model systems

Keywords: Large Language Model Security; Vertex AI; VPC Service Controls; Identity and Access Management Deny Policies; Private Service Connect; Model Context Protocol; Security Logging

1 Introduction

Cloud-hosted artificial intelligence systems are becoming foundational to modern enterprise platforms, especially in developer productivity, knowledge management, and software delivery workflows [1, 9]. Organizations are increasingly deploying internal large language model agents to perform tasks such as reviewing digital artifacts, summarizing technical content, recommending fixes, and automating repetitive workflows, code review with GitHub pull requests is one representative example of these patterns [1, 9]. However, the introduction of autonomous or semi-autonomous agents into cloud environments expands the attack surface beyond traditional web and infrastructure security concerns

* Corresponding author: Ranjan Kathuria

[1, 9]. In addition to classic cloud risks, these systems must now defend against prompt injection, data leakage through model interactions, token misuse, excessive tool permissions, incomplete observability, and unsafe automated actions [1, 9].

Google Cloud provides several native security capabilities for securing large language model systems, including Vertex AI for managed model inference, Private Service Connect for private network connectivity to Google-managed services, VPC Service Controls for placing Google APIs and services inside an API-level security perimeter that reduces data-exfiltration risk, centralized Identity and Access Management with Deny policies for enforcing non-bypassable access guardrails, Secret Manager for credential protection, and Cloud Logging and Cloud Monitoring for collecting audit, access, trace, and performance data [3-8]. For observability, the architecture enables request and response logging routed to BigQuery for structured querying and long-term retention, along with trace logging for model calls and tool invocations exported to a security information and event management platform [4]. Yet these features must be deliberately combined into a coherent architecture. If not, an internal large language model agent may gain excessive access to source code, secrets, or administrative functions, its network traffic may flow over uncontrolled paths, and its activities may go insufficiently monitored, creating a high-impact path for abuse or accidental exposure [1, 4, 5, 6].

Regarding VPC Service Controls specifically: in this architecture, VPC Service Controls are enabled by creating a service perimeter at the Google Cloud organization or project level, adding the Vertex AI API and related services as restricted services within that perimeter, and defining ingress and egress rules that allow only authorized identities and networks to access those services [3, 4]. The control is effective against Google API-channel exfiltration but does not govern raw outbound HTTPS calls to external systems such as third-party tool platforms; the latter is addressed through egress-proxy controls and firewall rules described in Section 4 [3, 4].

1.1.Problem Statement

Despite advances in managed large language model platforms, secure enterprise deployment of large language model agents remains challenging [1, 9]. Vertex AI can serve as a powerful model layer for internal agents, but without restrictive network, identity, tool, and observability boundaries, the platform can still expose sensitive prompts, proprietary data, inference outputs, and downstream tool interactions to misuse [3, 4, 9]. In any enterprise workflow where a large language model agent drives tool interactions, the agent may require limited access to external resources; however, it should not have the ability to perform destructive operations, modify workflows, access secrets, or execute unrelated administrative actions [1, 5, 6, 9].

A common design mistake is to rely primarily on allow policies and application logic while leaving broad outbound access, overly permissive service accounts, unrestricted third-party API paths, and missing or incomplete audit logging in place [5, 6]. This increases the probability of exfiltration, privilege misuse, and undetected compromise [5, 6]. A stronger approach is to design the platform so that sensitive resources are constrained by default, high-risk actions are denied even when broad roles are granted elsewhere, and all model interactions, tool invocations, and network flows are logged with sufficient detail to support detection and forensic investigation [3-6].

Objective

The objective of this research is to develop a secure-by-design reference architecture for an internal large language model agent platform on Google Cloud Platform using Vertex AI as the model layer and a Model Context Protocol based integration for controlled tool access [4, 7, 8]. Code review with GitHub is used as a running example, but the architecture is designed to generalize to other enterprise tool integration patterns [1, 9]. The architecture aims to reduce data-exfiltration risk, prevent privilege escalation, constrain unsafe agent actions, enforce observability through request and response logging, trace logging, and security information and event management integration, and provide measurable evidence of improved security posture through quantitative risk scoring [3-9].

2 Literature survey

Enterprise adoption of large language model platforms has accelerated significantly, with organizations deploying model-backed agents across workflows such as content review, artifact analysis, automated triage, and developer tooling; code review and pull request analysis represent one well-documented instantiation of this pattern [1, 9]. Research and platform guidance consistently identify data exfiltration, over-privileged identities, prompt injection, and insufficient observability as major risk categories for large language model deployments in cloud environments [1, 9]. These risks increase when agents are granted tool access to external systems, because model outputs begin to drive real-world actions with varying degrees of consequence [1, 9].

VPC Service Controls are positioned as a primary mechanism for reducing data-exfiltration risk across Google Cloud services, including Vertex AI, by enforcing an API-level service perimeter around selected projects and services [3, 4]. When the Vertex AI API is added as a restricted service within a perimeter, training data, model artifacts, online inference requests, and related assets are treated as protected resources that can be constrained to authorized networks and identities [3, 4]. This makes VPC Service Controls suitable for enterprise large language model environments where sensitive prompts and outputs are processed at scale, although the control applies to Google API traffic and not arbitrary outbound HTTPS calls to third-party systems [3, 4].

Identity and Access Management Deny policies operate as a layer distinct from allow policies in Google Cloud's permission evaluation model, and deny rules supersede conflicting allow grants [5, 6]. This makes them appropriate for enforcing non-bypassable guardrails on critical operations such as project deletion, policy modification, or log-sink tampering, even when principals hold broad inherited permissions [5, 6]. For private connectivity, Private Service Connect interface is recommended for bridging Google-managed runtimes such as Vertex AI Agent Engine to customer Virtual Private Cloud networks without traversing the public internet, and Domain Name System peering is part of that model for private resolution paths [7, 8].

Observability has become a core control domain for production artificial intelligence platforms because request and response logging, audit logging, trace capture, and centralized export to a security information and event management platform are necessary to detect prompt injection, anomalous tool use, and attempted exfiltration [4]. Together, network isolation, identity guardrails, tool scoping, and centralized observability form a layered platform-security model in which exfiltration paths, administrative misuse, unrestricted tool access, and undetected compromise are addressed before application-level safeguards are relied upon [3-7].

3 Problem definition or experimental work

3.1 Use Case Definition

This study examines an internal enterprise large language model agent platform deployed on Google Cloud Platform. The system uses Vertex AI as the inference layer and exposes a Model Context Protocol based tool interface that interacts with external enterprise systems; code review with GitHub pull requests serves as a concrete running example, but the architecture is intended to generalize to other tool-driven workflows [4, 7-9]. The agent receives task context such as artifact metadata, content diffs, or structured queries, analyzes the input using a Vertex AI-hosted model, generates recommendations or structured outputs, and invokes allowed tools to act on those outputs [4, 7, 9]. Because such workflows involve proprietary data, privileged credentials, and automated actions with real-world consequences, the platform must be secured against both external compromise and internal misuse [1, 5, 6].

3.2 Threat Model

The primary threats considered in this study are:

- Exfiltration of prompt content, proprietary data, or inference outputs from Vertex AI resources, exploiting gaps in service-perimeter configuration or unrestricted Google API access paths [3, 4].
- Unauthorized use of service-account or administrative permissions in the Google Cloud Platform project, including privilege escalation via overly broad Identity and Access Management allow policies not constrained by Deny policies [5, 6].
- Unsafe agent actions against external tools, such as merges, workflow edits, or repository modifications beyond the explicitly permitted non-destructive operations defined in the Model Context Protocol tool interface [1, 9].
- Secret leakage involving tokens for external tool platforms or internal service credentials [4].
- Unrestricted outbound access from the agent runtime to arbitrary external destinations [3, 7].
- Incomplete, disabled, or tampered logging and monitoring, where audit logs, access logs, request and response logs, or log-sink configurations are missing, insufficiently retained, or modified by a compromised identity [4-6].
- Tampering with security infrastructure controls, including unauthorized modification or deletion of VPC Service Controls perimeter policies, Identity and Access Management Deny policies, log sinks, or audit-log configurations [3-6].

3.3 Security Evaluation Formula

To assess the proposed design, this paper applies a quantitative risk-scoring approach consistent with established cybersecurity risk-evaluation methodology and prior secure-infrastructure evaluation practice [5, 6]. The formula used is:

$$RiskScore = Likelihood \times Impact$$

Likelihood represents the estimated probability that a threat scenario will occur under a given architecture, informed by the presence or absence of preventative controls and the exposure of the platform [5, 6]. Impact represents the severity of the business or technical consequence if the scenario occurs, including data loss, unauthorized automated actions, regulatory exposure, or degraded detection capability [5, 6].

3.4 Parameters For Calculation

Likelihood is estimated based on architectural exposure, access paths, identity configuration, and the presence or absence of controls such as VPC Service Controls perimeter enrollment, Identity and Access Management Deny coverage, Secret Manager restrictions, egress-proxy enforcement, and Model Context Protocol tool allowlisting [3-7]. Impact is rated on a scale of 1 to 10 based on data sensitivity, external-tool scope, administrative control surface, reversibility of agent actions, and operational or reputational damage [5, 6]. For logging and monitoring controls specifically, impact also reflects the loss of detection and forensic capability that occurs when audit logs, request and response logs, or log sinks are absent or tampered with [4-6].

4 Security infrastructure design

4.1 Core Security Components

The proposed design includes the following security controls, each addressing one or more of the threat categories identified in Section 3.2:

VPC Service Controls perimeter enclosing the Google Cloud Platform project that hosts the large language model agent, Vertex AI endpoints, Cloud Storage, BigQuery, and Secret Manager [3, 4]. The Vertex AI API and co-located Google-managed services are added as restricted services, and ingress and egress rules are defined to permit only authorized identities and networks [3, 4]. The project must be enrolled in the perimeter before any agent workload is deployed, as retroactive enrollment does not automatically secure previously provisioned resources [4].

Private Service Connect interface providing a private network bridge between the Google-managed Vertex AI Agent Engine runtime and the customer Virtual Private Cloud [7, 8]. All traffic between the agent runtime and internal services such as the Model Context Protocol server, internal APIs, or private databases flows over Google's internal network through Private Service Connect endpoints with Domain Name System peering [7, 8].

Egress proxy and firewall rules controlling all outbound traffic from the agent runtime to external systems [3, 7]. Because VPC Service Controls govern Google API traffic only, raw HTTPS calls to external destinations are controlled through a dedicated egress proxy and restrictive Virtual Private Cloud firewall egress rules that allowlist only the specific external endpoints the agent is permitted to reach [3, 7].

Identity and Access Management Deny policies applied at the organization, folder, or project level to block high-risk permissions regardless of broad allow-role grants [5, 6]. Deny policies are configured to prevent project deletion, modification of Identity and Access Management policies, disabling of audit logging, deletion or modification of log sinks, and removal of VPC Service Controls perimeter configuration [5, 6].

Model Context Protocol tool allowlisting so that the large language model agent can only invoke a narrowly defined set of non-destructive tools against external platforms [1, 9]. For the code-review example, this means the agent can read artifact diffs, retrieve metadata, and post review comments, but cannot perform merges, trigger workflows, modify repository settings, access secrets, or execute any other high-impact operations [1, 9].

Secret Manager with customer-managed encryption keys for storing all external-tool credentials, service-account keys, and integration tokens [4]. Secrets are accessible only from within the VPC Service Controls perimeter by explicitly authorized service accounts, with all access recorded in Cloud Audit Logs [4].

Centralized observability including Cloud Audit Logs for all Identity and Access Management decisions and administrative actions, request and response logging for Vertex AI model calls routed to BigQuery for long-term retention, trace logging for Model Context Protocol tool invocations and agent decision steps, Virtual Private Cloud Flow Logs for network-level visibility, and log-sink configurations protected by Identity and Access Management Deny policies [4, 5, 6]. All security-relevant events are exported to a security information and event management platform for correlation, alerting, and forensic investigation [4].

4.2 Proposed Google Cloud Platform Architecture

The secure design begins with an external tool platform or enterprise integration layer that emits events when artifacts change; a webhook from a source-code hosting system delivering pull-request metadata is used as the running example [1, 9]. These events are delivered to a Google Cloud Platform hosted controller service that validates the incoming payload, minimizes the context passed downstream, and strips any sensitive data not required for the large language model task. All inbound events from external platforms pass through a web application firewall and an HTTPS-enforced ingress endpoint before reaching the controller service, providing an initial layer of traffic inspection and rate limiting [3].

The controller service runs inside a project that is already enrolled in a VPC Service Controls perimeter before deployment, so that all interactions with Vertex AI and co-located Google-managed services are governed by the perimeter's ingress and egress rules [3, 4]. The controller invokes Vertex AI large language model endpoints using private Google API access paths within the perimeter, and model interactions, inference outputs, and associated artifacts are treated as protected assets that cannot leave the perimeter boundary through Google API channels unless permitted by explicit egress rules [3, 4]. Where the agent must communicate with private services in the customer Virtual Private Cloud, such as an internally hosted Model Context Protocol server or private internal APIs, that communication is established over a Private Service Connect interface with Domain Name System peering, so that traffic flows through private endpoints on Google's internal network rather than the public internet [7, 8].

The Model Context Protocol server acts as the policy-enforcement point for external-tool interactions, exposing a narrowly scoped set of tools such as reading artifact diffs, retrieving metadata, and posting non-destructive comments, while excluding high-risk operations such as merges, workflow modifications, administrative changes, pipeline triggers, or secret access [1, 9]. Outbound calls to external-tool APIs are routed through an egress proxy and firewall-controlled path that allowlists only the required external endpoints [3, 7]. All external-tool credentials and integration tokens are stored in Secret Manager with customer-managed encryption keys and are accessible only by explicitly authorized service accounts within the perimeter [4].

Identity and Access Management Deny policies apply cloud-side guardrails at the project or folder level, blocking project deletion, modification of sensitive Identity and Access Management bindings, disabling of Cloud Audit Logs, deletion of log sinks, and removal of VPC Service Controls perimeter configuration, even when principals hold broad inherited roles [5, 6]. All model calls, tool invocations, network flows, Identity and Access Management decisions, and secret-access events are captured through Cloud Audit Logs, request and response logging in BigQuery, and trace logs, with all security-relevant streams exported to a security information and event management platform for real-time correlation and long-term retention [4].

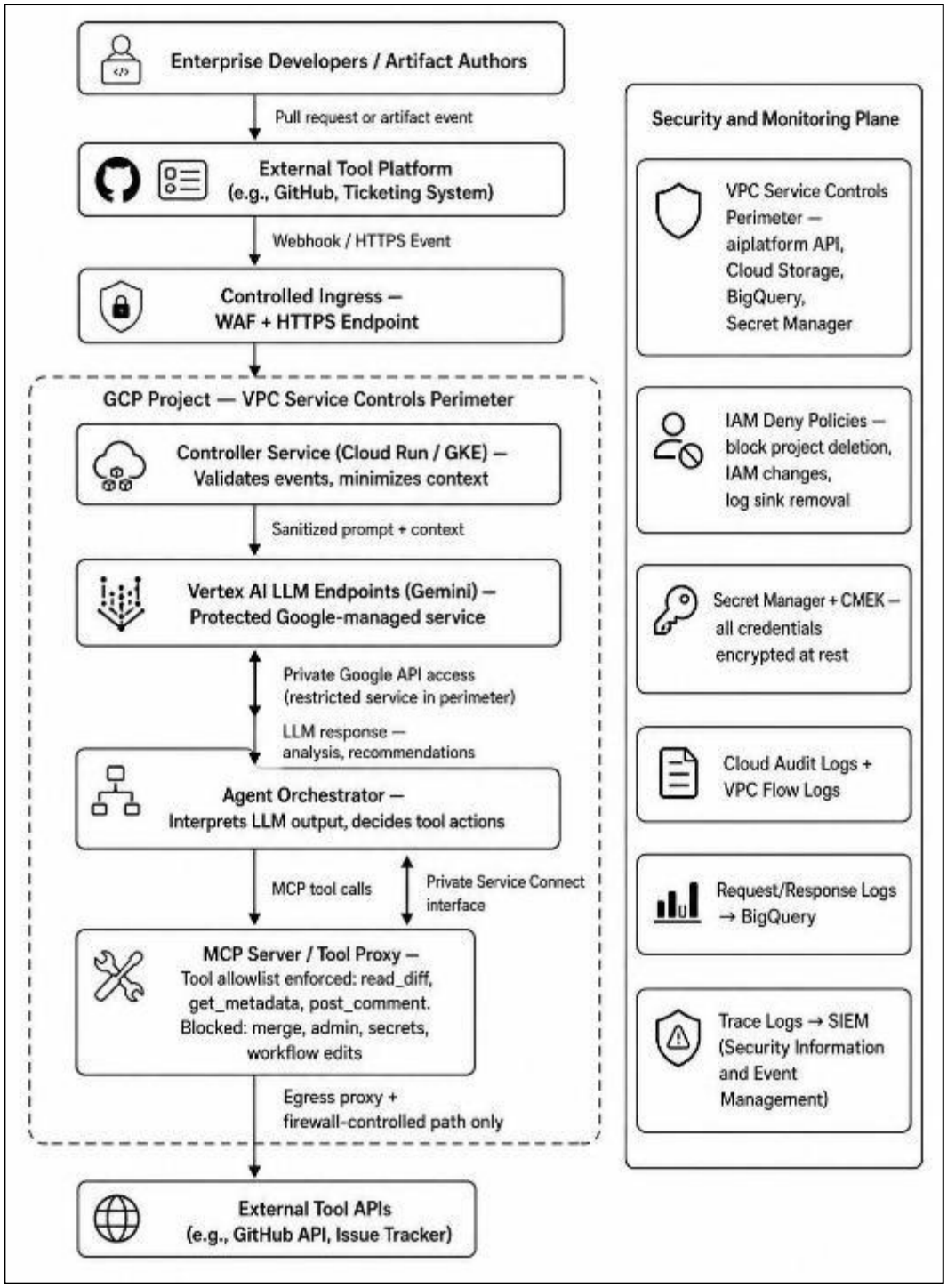

Figure 1 Secure Large Language Model Agent Platform on Google Cloud Platform

5 Analysis of proposed design and results

5.1 Component Level Risk Mitigation

The effectiveness of the architecture is evaluated through six control areas: VPC Service Controls perimeter, Identity and Access Management Deny policies, Private Service Connect interface with restricted egress, Model Context Protocol tool allowlisting, Secret Manager with encryption, and centralized logging and monitoring [3, 4, 5, 6, 7, 8, 9]. Each control is mapped to a specific risk vector identified in Section 3.2 and evaluated using the likelihood × impact model established in Section 3.3.

Table 1 Risk Evaluation - Pre & Post Mitigation

Control	Risk Reduction Mechanism	Pre-Mitigation Risk	Post-Mitigation Risk
VPC Service Controls Perimeter [3, 4]	Prevents protected Vertex AI artifacts and inference data from leaving the service perimeter through Google API channels	0.45 × 10 = 4.50	0.08 × 5 = 0.40
Identity and Access Management Deny Policies [5, 6]	Blocks use of sensitive permissions including project deletion, policy changes, and log-sink removal, even when broad allow roles exist	0.35 × 9 = 3.15	0.05 × 4 = 0.20
Private Service Connect Interface + Restricted Egress Proxy [7, 8]	Eliminates uncontrolled direct outbound traffic; internal service communication flows over private endpoints and external traffic through firewall-controlled egress proxy	0.30 × 8 = 2.40	0.07 × 4 = 0.28
Model Context Protocol Tool Allowlist + Action Gate [1, 9]	Restricts the agent to a narrowly scoped set of non-destructive tool actions; prevents unsafe autonomous operations such as merges, workflow modifications, or administrative changes	0.40 × 9 = 3.60	0.10 × 4 = 0.40
Secret Manager + Customer-Managed Encryption Keys [4]	Reduces token-exposure and hardcoded-credential risk; secrets are accessible only to authorized service accounts within the protected perimeter	0.35 × 10 = 3.50	0.05 × 4 = 0.20
Centralized Logging, Audit Logs, and Security Information and Event Management Export [4, 5, 6]	Reduces risk of undetected compromise, missing forensic evidence, and tampered security controls by retaining and exporting all model, network, and access telemetry	0.40 × 9 = 3.60	0.08 × 4 = 0.32

5.2 Systemic Risk Reduction

The total pre-mitigation risk score across all six control areas is 20.75 (4.50 + 3.15 + 2.40 + 3.60 + 3.50 + 3.60) [3-9]. The total post-mitigation risk score is 1.80 (0.40 + 0.20 + 0.28 + 0.40 + 0.20 + 0.32). The composite post-mitigation risk across the six evaluated control areas is therefore 1.80 / 6 = 0.30. This indicates a substantially reduced residual-risk profile after the preventative infrastructure, identity, networking, and observability controls are applied [3-6].

5.3 Risk Reduction Percentage

The reduction percentage is calculated as:

$$(1 - 1.80 / 20.75) \times 100 = 91.33\%$$

This result indicates that the proposed architecture reduces modeled platform risk by approximately 91.33% [3, 4, 5, 6, 7, 8, 9]. VPC Service Controls provide the strongest reduction against exfiltration-oriented threats by enforcing an API-level perimeter around Vertex AI and co-located services [3, 4]. Identity and Access Management Deny policies materially reduce residual risk from privileged misuse and silent guardrail bypass [5, 6]. Centralized logging and monitoring, protected by Deny policies against tampering, ensure that the risk reduction achieved by other controls is observable, verifiable, and defensible through retained audit evidence [4-6].

6 Conclusion

Secure-by-design principles are essential for enterprise large language model systems, especially when models are granted access to proprietary data, privileged credentials, and external tools that execute real-world actions. The architecture proposed in this paper demonstrates that security is achieved not by trusting the agent's outputs or relying on application-layer filtering alone, but by constraining what the agent, its runtime, its identities, and its tool interfaces are permitted to do at the infrastructure and identity layers before any model interaction occurs.

Vertex AI, when deployed inside a VPC Service Controls perimeter with relevant Google-managed APIs enrolled as restricted services, can be materially hardened against data exfiltration through Google API channels. Identity and Access Management Deny policies add a non-bypassable layer of control over privileged operations, ensuring that security infrastructure such as log sinks, perimeter configuration, and audit-log settings cannot be silently removed even by principals with broad inherited roles. Private Service Connect interface and egress-proxy controls ensure that agent network traffic flows only through private, inspectable, and firewall-governed paths. Model Context Protocol tool allowlisting limits agent behavior to a narrowly defined set of non-destructive actions, and centralized observability through audit logging, request and response logging in BigQuery, trace capture, and security information and event management export provides the retained evidence necessary for detection, incident response, and quantitative assurance.

The quantitative evaluation across six control areas shows that the proposed architecture reduces modeled platform risk by approximately **91.33%**, supporting the conclusion that enterprise large language model security should be designed around infrastructure-enforced boundaries, identity guardrails, controlled networking, minimal-action tool interfaces, and verifiable observability rather than model behavior alone. Future work may extend this methodology to multi-agent architectures, retrieval-augmented generation platforms handling sensitive data, and autonomous agent pipelines where the reversibility of actions and the blast radius of misuse are significantly higher.

References

[1] AI Agent Security. Securing the Model Context Protocol. Zenity. https://zenity.io/blog/security/securing-the-model-context-protocol-mcp Accessed 02 May 2026.

[2] [2] Binadox. Securing GCP with Vertex AI VPC Service Controls. https://www.binadox.com/blog/binadox-article-use-vpc-service-controls-for-vertex-ai/ Accessed 02 May 2026.

[3] [3] Google Cloud. VPC Service Controls. https://cloud.google.com/security/vpc-service-controls Accessed 02 May 2026.

[4] Google Cloud. VPC Service Controls with Vertex AI. https://docs.cloud.google.com/vertex-ai/docs/general/vpc-service-controls Accessed 02 May 2026.

[5] Google Cloud Blog. Introducing IAM Deny. https://cloud.google.com/blog/products/identity-security/introducing-iam-deny Accessed 02 May 2026.

[6] Google Cloud IAM. Deny access to resources. https://docs.cloud.google.com/iam/docs/deny-access Accessed 02 May 2026.

[7] Google Cloud Vertex AI. Set up a Private Service Connect interface for Agent Platform resources. https://docs.cloud.google.com/vertex-ai/docs/general/vpc-psc-i-setup Accessed 02 May 2026.

[8] Google Developer Forums. Vertex AI Agent Engine Networking Overview. https://discuss.google.dev/t/vertex-ai-agent-engine-networking-overview/267934 Accessed 02 May 2026.

[9] Trend Micro. Use VPC Service Controls for Vertex AI. https://www.trendmicro.com/trendaivisiononecloudriskmanagement/knowledge-base/gcp/VertexAI/use-vpc-service-controls-for-vertex-ai.html Accessed 02 May 2026.