CSAI Foundation

Cloud Security Alliance AI Safety Initiative

Vertex AI Service Agents as Lateral Movement Vehicles

Privilege Escalation Risks in Managed AI Platform Default Configurations

Unofficial AI-assisted Research

2026-04-01

Key Takeaways

- Multiple independent research teams have documented attack paths in Google Cloud's Vertex AI platform in which users holding read-only "Viewer"-level permissions can extract service agent credentials from GCP's instance metadata service and obtain project-wide access to Cloud Storage, BigQuery, Pub/Sub, and Artifact Registry resources.
- As of the January 2026 disclosure, Google classified the two vulnerabilities discovered by XM Cyber — affecting the Vertex AI Agent Engine and Ray on Vertex AI — as "working as intended," indicating no remediation was planned at that time. Organizations should verify current patch status against their deployed Vertex AI configuration, as the vulnerability landscape may have changed since initial disclosure. Absent a confirmed vendor patch, responsibility for mitigation falls on customers.
- A separate Unit 42 research disclosure in March 2026 demonstrated that a deployed Vertex AI agent using the Agent Development Kit (ADK) could harvest Per-Project, Per-Product Service Agent (P4SA) credentials from the metadata service and use them to pivot into tenant project infrastructure, accessing restricted Artifact Registry repositories and triggering pickle deserialization risks.
- The November 2024 "ModeLeak" research demonstrated a complete chain from a single `aiplatform.customJobs.create` permission to LLM model exfiltration via GKE lateral movement using IAM Workload Identity Federation. Google patched those specific vulnerabilities, but the architectural pattern remains.
- The core design tension is structural: Vertex AI service agents require broad default permissions to function seamlessly, and that breadth converts any compromise or misuse of the AI workload execution environment into a project-wide privilege escalation incident.
- Organizations should immediately audit all Vertex AI service agent identities, implement Bring Your Own Service Account (BYOSA) to replace default agents, and monitor metadata service access from AI workload nodes as a high-fidelity lateral movement indicator.

Background

Google Cloud's Vertex AI platform provides managed infrastructure for deploying, training, and serving machine learning models and AI agents. To support these workloads, the platform relies on a category of managed identities called service agents — Google-managed service accounts that are automatically created and attached to resources when AI workloads are provisioned. These identities operate invisibly in the background, performing platform operations such as reading model artifacts, writing to Cloud Storage, interacting with BigQuery, and managing container images in Artifact Registry. Their broad default permissions are a deliberate design choice: they ensure that Vertex AI features work out of the box without requiring operators to manually configure fine-grained IAM grants [1].

The platform exposes several distinct service agent types depending on the workload. The AI Platform Custom Code Service Agent (`service-<PROJECT_NUMBER>@gcp-sa-aiplatform-cc.iam.gserviceaccount.com`) is attached to custom training jobs, pipeline runs, and Ray clusters. The Reasoning Engine Service Agent handles agentic workloads deployed through Agent Engine. The Per-Project, Per-Product Service Agent (P4SA) is provisioned at the project level and serves as the primary identity used by AI agents built with the Agent Development Kit (ADK) [2][3]. Each of these identities carries permissions scoped to the entire project rather than to the individual workload, creating a permission model in which compromising or misusing any one identity grants access far beyond what any individual job or agent should require.

This structural characteristic has drawn sustained scrutiny from security researchers. Unit 42 published the first major disclosure, "ModeLeak," in November 2024, documenting how an attacker with a single `aiplatform.customJobs.create` permission could hijack the Custom Code Service Agent's access token from the metadata service and use it to traverse into a customer's full GCS bucket contents, BigQuery datasets, and — through GKE IAM Workload Identity Federation — neighboring Kubernetes clusters hosting deployed model containers [4]. In January 2026, XM Cyber disclosed two additional attack vectors targeting the Vertex AI Agent Engine and Ray on Vertex AI, both leveraging the same metadata service credential-exposure pattern. Most recently, in March 2026, Unit 42's "Double Agents" research demonstrated that an ADK-based agent deployed through Agent Engine could be weaponized to exfiltrate not only customer data but Google's own internal Artifact Registry contents [5].

Across all three research bodies, the common element is GCP's instance metadata service at `metadata.google.internal/computeMetadata/v1/instance/`. Any code executing within a Vertex AI compute environment — whether a custom training job, a Ray cluster head node, or a deployed Agent Engine reasoning engine — can query this endpoint without authentication and retrieve

the attached service agent's OAuth 2.0 access token. Because service agents are project-scoped and carry broad permissions, that token grants access to cloud resources well outside the workload's intended operational boundary [4][5][6].

Security Analysis

The Viewer-to-Service-Agent Escalation Path

XM Cyber's January 2026 research provides the most direct demonstration of low-privilege-to-high-privilege escalation in the Vertex AI platform: a user holding only the read-only "Vertex AI Viewer" role can achieve project-wide cloud access without exploiting any software vulnerability. The attack requires no code injection, no network exploit, and no lateral movement through a vulnerability chain — it requires only the ability to interact with existing Vertex AI console features.

In the Ray on Vertex AI attack path, the Vertex AI Viewer role grants two permissions — `aiplatform.persistentResources.list` and `aiplatform.persistentResources.get` — that are sufficient to access the "Head Node Interactive Shell" link in the Google Cloud Console. Despite being exposed to read-only users, this link provides root shell access to the Ray cluster's head node. From that position, a user queries the instance metadata service to retrieve the OAuth 2.0 access token for the Custom Code Service Agent. That token carries `devstorage.full_control` scope, granting read and write access to all Cloud Storage buckets in the project, along with access to BigQuery, Pub/Sub, and broad read access across the Google Cloud Platform [6][7].

The Agent Engine attack path differs slightly in mechanism but arrives at the same destination. An attacker with an existing reasoning engine can update it to include a malicious tool — for example, a standard-looking Python function that embeds a reverse shell. When the reasoning engine executes, the attacker gains code execution within the Agent Engine compute environment and queries the metadata service to extract the Reasoning Engine Service Agent's access token. Depending on the configuration, this token may grant access to the same broad project resources [7][8]. As of the January 2026 disclosure, Google's public response to both findings was that the services were "working as intended" [7]. Organizations should independently verify whether remediation has been issued for their Vertex AI version prior to treating these escalation paths as definitively unpatched.

The Double Agents Scenario: P4SA Credential Exfiltration

Unit 42's March 2026 "Double Agents" research extended the threat model to the agentic workload execution context. Researchers deployed a legitimate-appearing AI agent using Google's Agent Development Kit and demonstrated that when any call is made to a deployed Vertex agent through Agent Engine, the metadata service returns not only the P4SA access token but also the GCP project identifier, the agent's identity, and the OAuth scopes attached to the host machine. An agent that has been compromised through a prompt injection, a supply chain attack on a tool, or insider access can exfiltrate these credentials without generating visible alerts and use them for further access [5].

With P4SA credentials in hand, Unit 42 achieved three distinct levels of unauthorized access. First, they gained unrestricted read access to all Cloud Storage buckets within the customer project, bypassing the tenant isolation that users would reasonably expect Agent Engine to enforce, given its role as a managed execution environment. Second, they accessed restricted Artifact Registry repositories that contained proprietary container images used internally by Vertex AI's Reasoning Engine — infrastructure that is part of Google's own producer-side systems, not the customer's tenant environment. Third, they retrieved tenant project artifacts including `Dockerfile.zip`, `code.pkl`, and `requirements.txt` files that constitute the deployment specifications for customer agents. The `code.pkl` file uses Python's pickle serialization format, which by design permits arbitrary code execution during deserialization — making it a particularly high-risk artifact if write access is obtained through the P4SA. An attacker who gains write access to these files could implant persistent backdoors into the agent's deployment artifacts [5].

The ModeLeak Chain: From One Permission to Model Theft

The November 2024 ModeLeak research documented what remains the most complete end-to-end attack chain. An attacker holding only the `aiplatform.customJobs.create` permission — a permission frequently included in developer IAM roles for AI-enabled GCP projects, though its exact prevalence varies by organization — can inject malicious code into a custom training job's container specification. When the job runs, the attacker's code executes under the Custom Code Service Agent's identity and extracts that agent's access token. This token provides read and write access to all project Cloud Storage buckets and BigQuery tables, plus the ability to enumerate service accounts [4].

The second phase of the chain exploits IAM Workload Identity Federation between GCP and GKE. By deploying a poisoned model and obtaining the `custom-online-prediction` service account's credentials, the attacker can enumerate GKE clusters attached to the project and obtain cluster credentials. From within those clusters, using `crictl` with the service account's authentication token, the attacker can pull container images from Artifact Registry — including fine-tuned LLM adapter files

stored in Cloud Storage buckets with identifiers beginning with "caip" [4]. These adapter files contain proprietary fine-tuning weights, and their extraction effectively constitutes LLM intellectual property theft without any direct access to model training infrastructure. Google patched the specific vulnerabilities enabling this chain following responsible disclosure [4][11], but the underlying pattern — service agent tokens enabling GKE lateral movement via Workload Identity Federation — represents an architectural risk that extends beyond the specific vulnerabilities addressed.

Why Abused Service Agents Are Hard to Detect

A significant complicating factor in responding to this threat class is its detection resistance. Because service agents are managed Google identities operating as normal platform components, their activity appears in Cloud Audit Logs as standard Vertex AI infrastructure operations. An attacker using stolen P4SA credentials to read Cloud Storage buckets generates log entries that are structurally identical to those generated by legitimate Vertex AI platform activity. Security operations teams monitoring for unusual service account behavior must distinguish between a service agent accessing storage as part of normal AI workload orchestration versus a service agent whose credentials have been harvested and are being used by a threat actor [9]. In the authors' assessment, most enterprise SIEM configurations are unlikely to have detection rules tuned to this behavioral pattern, given that service agent activity closely mimics legitimate platform operations; cloud providers' own anomaly detection systems may similarly not flag such activity as suspicious if it falls within the normal operational envelope of the service agent's defined role [1][9].

Rock Lambros, CEO of RockCyber, a cloud security advisory firm, characterized the broader dynamic: "Cloud providers have turned 'shared responsibility' into a liability shield for their own insecure defaults." [1] Sanchit Vir Gogia of Greyhound Research observed: "Managed service agents are granted sweeping permissions so AI features can function out of the box. But that convenience comes at the cost of visibility and control." [9] Both perspectives reflect the underlying structural tension: the design choices that make Vertex AI operationally convenient are the same choices that create persistent, difficult-to-detect privilege escalation risk.

Recommendations

Immediate Actions

The highest-priority action is to replace default Vertex AI service agents with dedicated, scoped service accounts using Google's Bring Your Own Service Account (BYOSA) capability. Rather than accepting the platform's default P4SA and Custom Code Service Agent assignments — which carry project-wide permissions — organizations should create purpose-built service accounts for each Vertex AI workload class, grant only the permissions that workload requires, and configure Vertex AI to use those accounts instead of the defaults. Google has updated its official documentation to explicitly recommend this approach [2][3]. For organizations processing sensitive data or operating production workloads, BYOSA is the architecturally correct configuration, not a temporary workaround.

Organizations should also immediately audit all service agents currently attached to Vertex AI resources. The audit should enumerate every `gcp-sa-aiplatform*.iam.gserviceaccount.com` account in the project, document the roles and permissions granted to each, and identify any that carry `devstorage.full_control`, BigQuery admin, or Pub/Sub admin roles. This audit creates the baseline from which over-permissioned agents can be identified and scoped down.

Any environment using Ray on Vertex AI should be evaluated immediately for the Viewer-to-shell escalation path. If business requirements do not demand that "Viewer" role holders access Ray cluster head nodes, the Console link should be restricted or the Ray cluster redesigned to prohibit interactive shell access from roles below a defined trust boundary.

Short-Term Mitigations

Metadata service access from AI workload compute nodes should be treated as a high-fidelity lateral movement indicator and monitored accordingly. While legitimate Vertex AI operations do require metadata service access for identity token retrieval, unexpected queries to `metadata.google.internal/computeMetadata/v1/instance/service-accounts/` from Agent Engine reasoning engines or Ray head nodes — particularly queries that are followed by Cloud Storage or Artifact Registry access from unusual source identities — may indicate credential harvesting in progress. Cloud Audit Log entries for `storage.objects.list` and `artifactregistry.repositories.list` by service agent identities should be baselined and alerted on for deviations.

OAuth 2.0 scopes attached to service agents should be reviewed and restricted to the minimum necessary for each workload type. The `devstorage.full_control` scope attached by default to the Custom Code Service Agent is almost never required for training jobs or pipeline runs; replacing it with `devstorage.read_only` or scoped bucket-level permissions eliminates the most impactful storage access vector while preserving functional operation. Where read-write storage access is genuinely required, it should be scoped to the specific buckets and prefixes the workload needs rather than the entire project [10].

Python pickle deserialization in `code.pkl` artifacts should be treated as an active code execution risk. Until Google transitions to a safer serialization format, organizations should treat any tenant-project artifact files as untrusted inputs, scan them for unexpected content during deployment pipelines, and implement file integrity monitoring on Agent Engine deployment directories.

Strategic Considerations

The structural issue exposed by this research series extends beyond Vertex AI to the broader category of managed AI platforms. When a cloud provider bundles managed identity with AI workload execution, the security boundary between the AI workload and the surrounding cloud environment is only as strong as the permissions attached to the managed identity. The structural pattern documented here — managed identity with broad defaults, exposed via metadata service — is architecturally present in analogous managed AI services across major cloud providers. Organizations using Amazon SageMaker or Azure Machine Learning should conduct equivalent audits of execution role and managed identity permissions, applying the least-privilege principles described above; however, the specific exploitation history and default permission scope documented in this note is specific to Vertex AI and should not be assumed to apply identically to other platforms without independent research.

Organizations building AI governance programs should establish a formal category for AI workload identity in their IAM governance frameworks. This includes periodic audits of service account permissions attached to AI infrastructure, explicit lifecycle management for managed identities (so that service agents from decommissioned workloads do not persist with live permissions), and privileged access management controls applied to any human or system identity capable of deploying or modifying AI workloads. The `aiplatform.customJobs.create` and `aiplatform.persistentResources.list` permissions demonstrated in this research to enable privilege escalation should be treated as high-sensitivity permissions requiring the same governance scrutiny as IAM admin roles.

Supply chain risk for AI models and agents requires specific attention in light of the ModeLeak research. An organization that inadvertently deploys a compromised model from a public registry — such as a poisoned Hugging Face model — may be granting an external attacker the `custom-online-prediction` service account identity and, through it, access to GKE clusters. Model provenance verification, cryptographic signing of model artifacts, and behavioral sandboxing of models in isolated compute environments prior to production deployment are the appropriate controls for this risk vector.

CSA Resource Alignment

This research note's findings map to several established Cloud Security Alliance frameworks and publications. The MAESTRO (Multi-Agent Environment, Security, Threat, Risk, & Outcome) threat modeling framework [12] addresses risks in the infrastructure and governance layer of agentic AI deployments, including overprivileged infrastructure identities, service account hygiene, execution environment isolation, and credential exposure through platform APIs. The service agent privilege escalation pattern documented here is a concrete instantiation of the threat category MAESTRO describes as privileged execution context abuse, in which AI workload execution environments provide a pathway to infrastructure access beyond the intended operational boundary.

The CSA Cloud Controls Matrix v4.1 [13] addresses this risk under the Identity and Access Management (IAM) control family, particularly IAM-04 (Separation of Duties), IAM-06 (Least Privilege), and IAM-09 (User Access Provisioning). The CCM's Infrastructure and Virtualization Security (IVS) domain, specifically IVS-04 and IVS-09, is relevant to the isolation failures between Agent Engine compute environments and tenant project resources. Under the AI and Machine Learning security domain, CCM AIS-03 (AI / ML Platform Supply Chain Risk Management) directly addresses the model poisoning vector documented in the ModeLeak research. Organizations deploying Vertex AI should use these control references as the basis for assessing their current configuration against the risk surface documented here.

The CSA AI Organizational Responsibilities (AIOR) framework's principle of "Continuous Runtime Monitoring" applies directly to the detection gap identified in this research: because abused service agents generate activity indistinguishable from normal platform operations, organizations cannot rely on perimeter controls alone and must instrument the runtime environment for behavioral anomaly detection. The AICM (AI Controls Matrix), as a superset of the CCM, extends these controls specifically to AI system deployment contexts and should be the primary reference for organizations building AI-specific IAM governance programs.

CSA's Zero Trust guidance is particularly relevant to the BYOSA recommendation. Zero Trust's principle of "never trust, always verify" applied to service identities means treating each AI workload as a distinct identity principal that must be explicitly authorized for specific resources rather than relying on platform-default permissions. This is architecturally equivalent to the BYOSA control: replacing Google-managed service agents with customer-managed, workload-scoped service accounts is a Zero Trust implementation, not merely a configuration preference.

Finally, the CSA Security Trust Assurance and Risk (STAR) registry provides a mechanism for cloud customers to assess Google Cloud's published security controls against these findings. Organizations procuring Vertex AI services should use STAR assessments to evaluate whether Google Cloud's control documentation adequately discloses the default service agent permission model and the customer responsibilities associated with securing it.

References

[1] Greyhound Research / Sanchit Vir Gogia. "Google Vertex AI security permissions could amplify insider threats". CSO Online, 2026.

[2] Google Cloud. "Use a custom service account | Vertex AI". Google Cloud Documentation, 2026.

[3] Google Cloud. "Use agent identity with Vertex AI Agent Engine". Google Cloud Documentation, 2026.

[4] Palo Alto Networks Unit 42. "ModeLeak: Privilege Escalation to LLM Model Exfiltration in Vertex AI". Unit 42 Threat Research, November 2024.

[5] Palo Alto Networks Unit 42. "Double Agents: Exposing Security Blind Spots in GCP Vertex AI". Unit 42 Threat Research, March 2026.

[6] XM Cyber. "Critical Privilege Escalation Vulnerabilities Discovered In Google Vertex AI". As reported by First Hackers News, January 2026.

[7] Cybersecurity News. "Google's Vertex AI Vulnerability Enables Low-Privileged Users to Gain Service Agent Roles". CybersecurityNews.com, January 2026.

[8] GBHackers. "Google Vertex AI Flaw Lets Low-Privilege Users Escalate to Service Agent Roles". GBHackers on Security, January 2026.

[9] RockCyber / Rock Lambros. Quoted in: "Google Vertex AI security permissions could amplify insider threats". InfoWorld, 2026.

[10] The Hacker News. "Vertex AI Vulnerability Exposes Google Cloud Data and Private Artifacts". The Hacker News, March 2026.

[11] SC Media. "Google fixes 2 Vertex AI flaws that could lead to privilege escalation, model leaks". SC Media, 2024.

[12] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO". CSA AI Safety Initiative, February 2025.

[13] Cloud Security Alliance. "Cloud Controls Matrix v4.1". CSA, 2023.

