Network Working Group
Internet-Draft
Intended status: Informational
Expires: 30 September 2026

K. Hartman
SANS Institute

29 March 2026

Credential Broker for Agents (CB4A)
draft-hartman-credential-broker-4-agents-00

Abstract

This document specifies a Credential Broker for Agents (CB4A). a credential vaulting and brokering architecture that mediates AI agent access to API credentials. Agents never hold real long-lived credentials. Instead, they receive short-lived, narrowly scoped, auditable proxy credentials issued by a broker that separates policy decisions from credential delivery. The architecture addresses the "credential sprawl" risk inherent in agentic AI, where agents aggregating access across many services become high-value compromise targets.

CB4A builds on SPIFFE/SPIRE for workload identity, uses a Policy Decision Point / Credential Delivery Point separation inspired by NIST SP 800-207, and employs DPoP (RFC 9449) for sender-constrained token binding. The specification defines three credential proxy models, a tiered approval framework, and a comprehensive threat model with ten identified threats and their mitigations.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 30 September 2026.

Table of Contents

1. Introduction

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Overview

AI agents increasingly need to call APIs on behalf of users. Current approaches give agents long-lived API keys or OAuth tokens, creating standing privileges that survive beyond the task, aggregate access across services, and make agents high-value compromise targets. A compromised agent (via prompt injection, context exfiltration, or tool-calling manipulation) inherits all of those credentials.

The Credential Broker for Agents (CB4A) addresses this by introducing a credential vaulting and brokering layer that sits between agents and the services they need to reach. The agent never holds real credentials. Instead, it receives short-lived, narrowly scoped, auditable proxy credentials.

1.3. Real-World Motivation

The credential sprawl risk for AI agents is not theoretical. In March 2026, the TeamPCP supply chain campaign compromised LiteLLM [TEAMPCP-LITELLM], an AI gateway proxy used by thousands of enterprises to route requests to LLM providers (OpenAI, Anthropic, Google Vertex AI, and others). LiteLLM’s entire purpose is to hold API keys for dozens of AI providers, making it one of the highest-density credential targets in any infrastructure. The attackers injected a credential stealer into PyPI packages that harvested SSH keys, cloud credentials, LLM API keys, .env files, and database passwords from any machine running the compromised versions [LITELLM-SECURITY]. The campaign exfiltrated an estimated 300+ GB of compressed credentials affecting approximately 500,000 corporate identities.

This incident illustrates precisely the problem CB4A addresses: when AI infrastructure concentrates long-lived credentials in a single process or configuration, compromise of that process yields catastrophic access.

1.4. Why a Credential Broker for Agents

The credential sprawl problem for agentic AI is distinct from traditional service-to-service authentication because:

* Agents are semi-autonomous: they make decisions about which APIs to call, unlike microservices with fixed call patterns.

* Agent compromise vectors are novel: prompt injection, context window exfiltration, and tool-calling manipulation do not exist in traditional service architectures.

* Scope is unpredictable: an agent might need Slack write access for one task and GitHub admin for the next, making static scoping insufficient.

* User delegation is implicit: agents act on behalf of users without the user being present for each action.

2. Architecture Overview

2.1. Design Philosophy

CB4A follows three architectural principles:

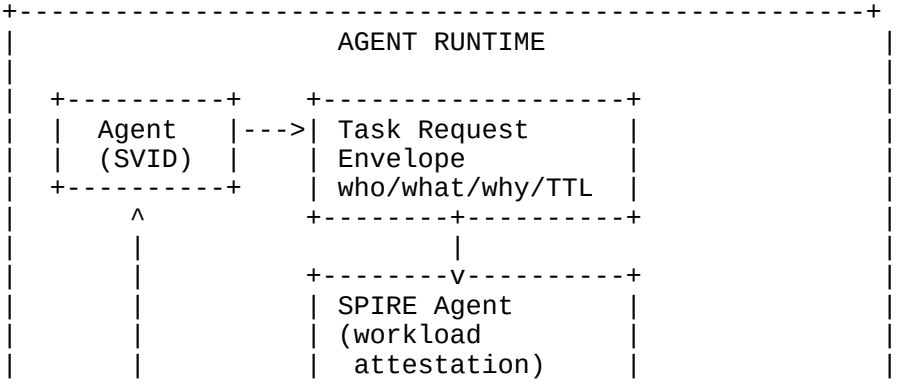
1. Build on SPIFFE/SPIRE, don't build bespoke: The identity, attestation, and certificate management problems are already solved. CB4A adds an agent-specific policy and credential mediation layer on top.

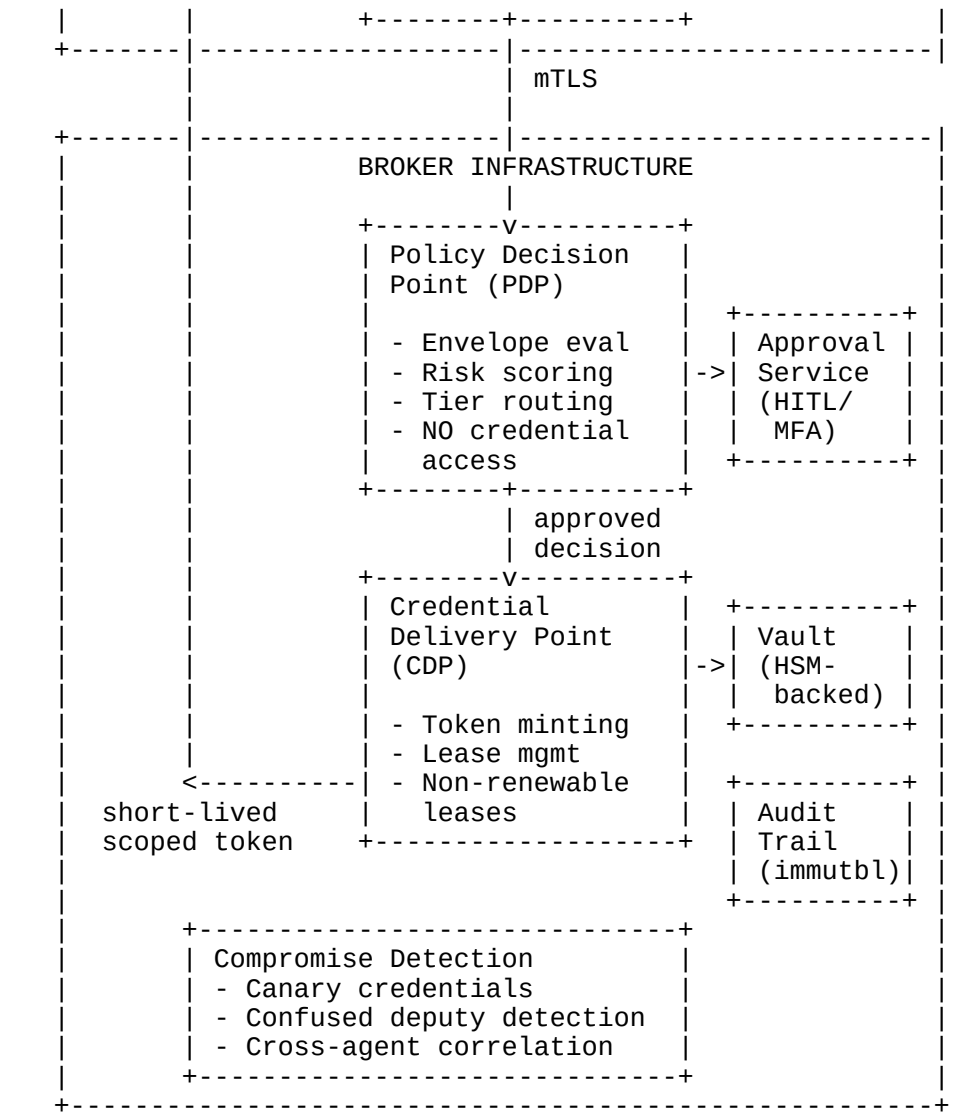
2. Separate policy from credentials: The component that decides "yes" (PDP) MUST never touch credential material. The component that dispenses credentials (CDP) MUST never make policy decisions. Compromise of one does not yield the other.

3. Envelope as evidence, not authorization: The Task Request Envelope is an auditable artifact that the agent produces. It is evaluated by the PDP, but the PDP's decision is based on independent policy evaluation and behavioral baselines (inspired by BeyondCorp [BEYONDCORP]), not on trusting the agent's self-reported justification.

2.2. System Architecture

The following diagram illustrates the CB4A system architecture with its trust boundaries:





2.3. Scalability Philosophy

CB4A is designed to scale from a single-machine home deployment to a distributed enterprise environment. The same conceptual architecture applies at every scale; what changes is how components are deployed, not what they do. A home user running one agent on a laptop and an enterprise running thousands of agents across data centers both benefit from the same PDP/CDP split, envelope-based evidence, and short-lived tokens. These are architectural properties, not

deployment choices.

2.4. Trust Boundaries

Boundary	Separates	Compromise of A yields
Agent / SPIRE Agent	Workload from identity	Agent identity (bounded by SVID TTL)
SPIRE Agent / PDP	Identity from policy	Ability to submit envelopes (not approve them)
PDP / CDP	Policy from credentials	Approval decisions (not credential material)
CDP / Vault	Delivery from storage	Active credentials (bounded by HSM access policies)
PDP / Approval Svc	Auto from human approval	Auto-approval only (HITL/MFA still requires human)

Table 1

2.5. Credential Flow

1. Agent constructs Task Request Envelope with SVID identity.
2. Envelope submitted to PDP via mTLS (SPIRE-attested).
3. PDP evaluates envelope against policy, routes to appropriate approval tier.
4. If approved, PDP sends signed approval decision to CDP.
5. CDP mints short-lived, scoped token with non-renewable lease.
6. Token returned to agent, bound to request context.
7. Agent uses token directly against target service.
8. Token expires; re-attestation required for new token.
9. Full chain logged to immutable audit trail.

3. Credential Proxy Models

CB4A defines three models for how the broker mediates credential access between an agent and a target service. These models represent different points on a tradeoff between security isolation, operational complexity, and compatibility with existing APIs.

3.1. Model A: Proxy Gateway

The agent never receives any real credential. Instead, the broker operates a proxy endpoint that the agent calls. The broker receives the agent's request, validates the agent's session token, injects the real API credential, forwards the request to the target service, and returns the response. The agent never receives, sees, or holds the real credential. The credential travels from the broker to the target service, but never to the agent.

```
Agent --request--> CB4A Proxy --request + real cred--> Target API
Agent <-response-- CB4A Proxy <-response------------- Target API
```

Strengths:

* Strongest isolation: the agent never sees, holds, or transmits a real credential.

* The broker can inspect, log, and filter every request and response in real time.

* Works with any target API without requiring the target to support short-lived tokens.

Weaknesses:

* Every API call becomes two network hops, adding latency.

* The proxy is a single point of failure.

* The proxy must understand the request/response format of every target API it mediates.

* At high scale, the proxy becomes a throughput bottleneck.

Best for high-security environments where credential isolation is paramount, or as a fallback for target services that do not support short-lived tokens natively.

3.2. Model B: Short-Lived Token Minting

The broker uses the real long-lived credential to mint a derivative short-lived token with narrow scope, then hands that short-lived token to the agent. The agent uses the short-lived token to call the target service directly. The token expires automatically after a brief TTL (seconds to minutes). The specific minting mechanism varies by platform.

```
Agent --req token--> CB4A Broker --mint--> Token Endpoint
Agent <-short-lived-- CB4A Broker

Agent --req + short-lived token--> Target API (direct)
Agent <-response----------------- Target API
```

Platform-specific minting mechanisms:

Platform	Mechanism	Description
AWS	Security Token Service (STS) AssumeRole	Mints temporary access keys with session policies and configurable TTL
Google Cloud	OAuth 2.0 + Service Account Impersonation	Uses generateAccessToken API to create short-lived OAuth 2.0 access tokens
Azure	Azure AD / Entra ID token endpoint	Issues short-lived OAuth 2.0 bearer tokens via client credentials flow
GitHub	App Installation Tokens	Generates short-lived installation access tokens scoped to repositories
Generic	OAuth 2.0 Token Exchange [RFC8693]	Exchanges long-lived token for short-lived, narrowly scoped derivative

Table 2

Strengths:

* No proxy bottleneck: the agent calls the target API directly.

* Short-lived tokens limit blast radius to the TTL window.

* Proven at massive scale across all major cloud providers.

* The broker only handles token minting, not ongoing proxying.

Weaknesses:

* The agent holds a real (if temporary) credential in memory.

* Requires the target service to support short-lived minting.

* Scope granularity depends on the target service's token API.

This is the RECOMMENDED primary model for CB4A.

3.3. Model C: Credential Wrapping with Scheduled Revocation

For target APIs that only support long-lived credentials and have no mechanism for minting short-lived tokens, the broker hands the agent the actual long-lived credential but immediately schedules its revocation after the task completes or the TTL expires.

Strengths:

* Works with any API regardless of token support.

* Simple to implement.

Weaknesses:

* The agent holds a real, long-lived credential during the window, the weakest isolation of all three models.

* Revocation depends on the target service's key management.

* If revocation fails, the credential remains live indefinitely.

This is the WEAKEST model and is NOT RECOMMENDED for new integrations. Use only for legacy APIs with no alternative.

3.4. Model Comparison

Property	Model A (Proxy)	Model B (Mint)	Model C (Wrap)
Agent holds real cred?	Never	Temporary	Yes (until revoked)
Latency	High (double	Low (one-time	None

overhead	hop)	mint)	
Single point of failure?	Yes (proxy)	No	No
Works with any API?	Yes	Short-lived support only	Yes
Blast radius	Minimal	Bounded by TTL	Full until revoked
Recommended use	DPoP fallback	Primary model	Legacy only

Table 3

3.5. CB4A Model Strategy

CB4A uses Model B as the primary credential model wherever the target service supports short-lived token minting. Model A is used selectively as a DPoP enforcement layer. Model C is available as a last resort for legacy integrations but is discouraged and flagged in audit logs.

4. Technical Design

4.1. SPIRE-Based Identity Layer

Agents receive identity through SPIRE [SPIRE] workload attestation. Each agent session gets a SPIFFE [SPIFFE] Verifiable Identity Document (SVID), a short-lived X.509 certificate or JWT.

SVID Properties:

* TTL: matches agent session lifetime (default: 1 hour, configurable).

* SPIFFE ID format: spiffe://trust-domain/agent/{user-id}/{session-id}

* Attestation: node attestation (platform identity) plus workload attestation (agent process identity).

* Non-renewable: agent MUST re-attest to get new SVID.

Trust Domain Federation: For multi-tenant or multi-organization deployments, SPIFFE federation enables cross-domain trust without sharing root CAs.

4.2. Task Request Envelope

The envelope is a structured, signed evidence artifact submitted by the agent for every credential request.

```
{
  "envelope_version": "1.0",
  "agent_svid": "<SPIFFE SVID reference>",
  "request_id": "<UUID>",
  "timestamp": "<ISO-8601>",
  "target": {
    "service": "slack",
    "action": "chat.postMessage",
    "resource": "#engineering",
    "scope": ["channels:write"]
  },
  "justification": {
    "task_id": "<originating user instruction ref>",
    "description": "Post weekly standup summary"
  },
  "ttl_seconds": 60,
  "signature": "<SVID-signed envelope hash>"
}
```

The justification.description field is auditable evidence, NOT an authorization input. The PDP MUST NOT evaluate justification text for approval decisions. Approval is based on: (1) agent identity (SVID), (2) target service/action/scope matching against policy, (3) behavioral baseline comparison, and (4) approval tier routing. The justification exists solely for post-incident forensic reconstruction.

4.3. Policy Decision Point (PDP)

The PDP is a stateless service that evaluates envelopes against policy, following the PDP/PEP separation defined in [NIST-ZTA]. It has zero access to credential material.

Inputs:

* Task Request Envelope

* Agent identity (SVID validation via SPIRE)

* Policy rules (loaded from versioned policy store)

* Behavioral baseline (historical access patterns)

Outputs:

* Signed approval decision (approved/denied/escalated)

* Approval tier (auto/HITL/MFA)

* Scope constraints (may narrow requested scope)

* Decision rationale (logged)

Scope Granularity: The PDP's scope enforcement is configurable by the administrator. Implementations may range from coarse service-level scoping (e.g., "Slack: read/write") to fine-grained resource-level scoping (e.g., "Slack: write to #engineering only"). The appropriate granularity depends on the deployment's threat model and operational maturity.

Policy evaluation MUST use a typed, compiled policy language (not string interpolation). Envelope fields MUST be deserialized into typed structures before evaluation. No string concatenation or template rendering in policy evaluation paths.

Tier	Criteria	Approval Flow
Tier 1 (Auto)	Low-risk, pre-authorized, within baseline	Immediate approval
Tier 2 (HITL)	Moderate risk, outside baseline	Async human approval via notification
Tier 3 (MFA)	High risk, destructive, admin-level	Synchronous approval with MFA challenge

Table 4

4.4. Credential Delivery Point (CDP)

The CDP brokers vault [VAULT] interactions and mints short-lived tokens. It accepts only signed PDP approval decisions.

Token Minting (Model B):

* CDP receives signed approval decision specifying exact scope and TTL.

* CDP retrieves base credential from HSM-backed vault (or encrypted local store for single-machine deployments).

* CDP mints a derivative short-lived token using the target service's native mechanism (AWS STS AssumeRole, GCP generateAccessToken, GitHub App installation tokens, OAuth 2.0 Token Exchange [RFC8693], etc.).

* For services without native short-lived token support: CDP issues a CB4A proxy token and operates a thin proxy endpoint (Model A fallback).

* Token is non-renewable; expiry is absolute, no refresh.

Lease Management:

* Every issued token has a lease tracked by the CDP.

* Leases are non-renewable by default.

* On lease expiry: token is invalidated, agent MUST re-attest.

* On anomaly detection signal: CDP can revoke active leases immediately.

The CDP MUST NOT cache decrypted credentials in memory beyond the minting operation. Credentials are fetched from vault, used to mint derivative token, and immediately zeroed. The CDP process SHOULD run in a hardened runtime with no shell access, restricted outbound network, and memory encryption where platform supports it.

4.5. Sender-Constrained Tokens

Ephemeral tokens in agent memory are extractable and replayable. CB4A addresses this with sender-constrained tokens using DPoP [RFC9449]:

* At token minting, agent generates an ephemeral key pair.

* Agent provides public key to CDP.

* CDP binds the minted token to the agent's public key.

* On each API call, agent signs the request with its private key (DPoP proof).

* Target service (or CB4A proxy for Model A fallback services) validates both the token and the DPoP proof.

* Stolen tokens are useless without the corresponding private key.

For services that do not support DPoP natively, the CB4A proxy endpoint validates DPoP proof before injecting real credential. This is the only scenario where Model A (proxy) is used, as a DPoP enforcement layer for services lacking native support.

4.6. Tiered Approval Service

Tier 1: Auto-Approval:

* PDP evaluates against static policy plus behavioral baseline.

* Approval latency: less than 10ms.

Tier 2: Human-in-the-Loop:

* Push notification to approver’s device.

* Approver sees: agent identity, requested action, scope, justification.

* Timeout: configurable (default 5 minutes), denied on timeout.

Tier 3: MFA-Required:

* Synchronous approval with multi-factor authentication.

* Approver MUST complete MFA challenge (FIDO2/WebAuthn, not SMS/TOTP).

4.6.1. Approval Routing (Enterprise)

In enterprise deployments, approval requests MUST be routed to the appropriate human reviewer, not a generic approval queue. The agent’s owner or responsible team is the correct approver because they understand the agent’s purpose, expected behavior, and legitimate access patterns.

Routing Criterion	Example
Agent owner	Agent sales-report-bot routes to Sales Eng team lead
Service owner	Billing API requests route to Finance team
Escalation chain	Timeout escalates to manager or backup
On-call rotation	After-hours routes to current on-call

Table 5

For home/individual deployments, approval routing is trivial. all requests route to the single user.

4.6.2. Approval Fatigue Countermeasures

* Rate limiting on approval requests per agent per time window.
* Anomaly flagging when approval request patterns change.
* Mandatory cool-down periods after N consecutive approvals.
* Periodic "challenge approvals" (known-benign requests presented as suspicious) to verify approver attention.

4.7. Immutable Audit Trail

Every interaction produces a structured, append-only log entry:

```
{
  "event_type": "credential_request|approval|issuance|
                 usage|expiry|revocation",
  "timestamp": "<ISO-8601>",
  "agent_spiffe_id": "spiffe://trust-domain/agent/user/session",
  "envelope_hash": "<SHA-256 of request envelope>",
  "decision": "approved|denied|escalated|timed_out",
  "decision_tier": "auto|hitl|mfa",
  "credential_scope": ["channels:write"],
  "credential_ttl_seconds": 60,
  "target_service": "slack",
  "target_action": "chat.postMessage",
  "approver_identity": "<human approver ID or 'auto'>",
  "correlation_id": "<links related events>"
}
```

Storage: Write-once storage (append-only log, S3 with object lock, or dedicated SIEM). Log pipeline failures MUST trigger fail-closed behavior; no credentials issued if audit trail is unavailable.

Audit trail MUST be written to a separate trust boundary from both PDP and CDP. Neither PDP nor CDP has write access to modify or delete existing log entries. Log integrity is verified via hash chaining.

4.8. Compromise Detection

Canary Credentials: Seed the credential store with tokens that should never be used. Any access to a canary triggers immediate alert and session termination.

Confused Deputy Detection: Monitor for credential requests where the agent identity and the task context do not match.

Cross-Agent Correlation: Monitor for coordinated multi-agent access patterns that individually appear benign but collectively constitute data exfiltration or privilege escalation.

Runtime Behavior Monitoring (Recommended): Runtime monitoring of post-issuance credential usage is highly valuable. Even a short-lived token can cause significant damage if misused. Implementations SHOULD include anomaly detection on API call patterns, mid-flight kill switches, and behavioral baselines.

4.9. Secure Degradation

CB4A specifies secure degradation behavior for every failure mode:

Failure	Behavior	Rationale
PDP unavailable	Fail-closed	Policy cannot be evaluated
CDP unavailable	Fail-closed	Credentials cannot be minted
Vault unavailable	Fail-closed	Base credentials inaccessible
Audit trail unavailable	Fail-closed	Cannot maintain chain of custody
Approval svc unavailable	Tier 1 continues; Tier 2/3 denied	Preserve low-risk operations
SPIRE unavailable	No new sessions; existing SVIDs valid	Identity cannot be attested

Table 6

Emergency Break-Glass: Requires two-person authorization (dual key). Bypasses PDP but still logs through audit trail. Automatically expires after configurable window (default: 30 minutes). Triggers mandatory post-incident review.

4.10. Broker Bypass Prevention

A credential broker is only effective if agents cannot circumvent it to access credentials directly. This is analogous to the bypass problem faced by Cloud Access Security Brokers (CASBs), which must prevent users and applications from reaching cloud services outside the broker's mediation.

CB4A draws three architectural lessons from CASB deployments:

4.10.1. Lesson 1: Agents Must Not Have Direct Credential Access

The most fundamental control: real credentials MUST be stored in a vault that agents cannot access directly. The agent runtime MUST NOT have network connectivity to the vault, IAM permissions to read secrets, or file system access to credential stores. Only the CDP has vault access. This is the CB4A equivalent of a CASB's network-level enforcement: if the agent cannot reach the credentials, it cannot bypass the broker.

4.10.2. Lesson 2: Network-Level Enforcement Over Agent Cooperation

CASB experience shows that agent-based enforcement (where the agent voluntarily routes traffic through the broker) is weaker than network-level enforcement (where the network prevents direct access). CB4A implementations SHOULD enforce broker mediation at the infrastructure level:

* Network policies (firewall rules, security groups, or Kubernetes NetworkPolicy) that block agent-to-service direct communication for brokered services.

* Service mesh sidecars that intercept outbound traffic and validate CB4A tokens before forwarding.

* DNS-level controls that resolve brokered service endpoints to the CB4A proxy rather than the real service.

Relying solely on the agent's cooperation to use the broker is insufficient; a compromised agent will attempt to call services directly if network-level controls do not prevent it.

4.10.3. Lesson 3: Multimode Detection

Even with network-level enforcement, sophisticated bypass attempts may succeed (e.g., tunneling through allowed endpoints). CASB architectures address this with API-based monitoring alongside inline enforcement. CB4A implementations SHOULD:

* Monitor target service audit logs for API calls that did not originate from the CB4A proxy or use CB4A-issued tokens.

* Alert on credential usage patterns that do not match CB4A audit trail records (indicating credentials were used outside the broker).

* Implement canary detection at the service level, not just at the vault level.

4.11. Native Integration Specification (Future)

If CB4A achieves sufficient adoption, target services could natively validate CB4A-issued credentials. CB4A-native tokens would be JWTs containing:

* iss: CB4A broker SPIFFE ID

* sub: Agent SPIFFE ID

* aud: Target service identifier

* scope: Requested permissions (service-specific format)

* exp: Expiry timestamp (short TTL)

* jti: Unique token ID (for replay detection)

* cnf: Confirmation claim binding token to DPoP key [RFC9449]

* envelope_hash: SHA-256 of the originating Task Request Envelope

Discovery: Target services would discover CB4A broker endpoints via a well-known URI (.well-known/cba-configuration) publishing the JWKS URL, supported scopes, and token format version.

5. Scalability Model

CB4A separates logical components from their deployment topology. The same security properties hold at every scale.

5.1. Home / Single-Machine Deployment

A single user running AI agents on a personal workstation. All CB4A components run as local processes or lightweight containers on the same machine. A minimal deployment uses three components: SPIRE (identity), PDP (policy), and a credential proxy that combines CDP and vault access.

The trust model is identical to enterprise, just deployed on one machine instead of across a network.

5.2. Small Team / Startup Deployment

A team of 5-50 people with multiple agents across workstations and CI/CD pipelines. Deploys the same three core components as a home deployment (SPIRE, PDP, credential proxy), with the option to separate the credential proxy into distinct CDP and vault integration services as the team grows.

5.3. Enterprise Deployment

Thousands of agents across multiple teams, data centers, and trust domains. Typically uses five or more components with dedicated infrastructure per component, including federated SPIRE, HSM-backed vault clusters, and full cross-agent correlation.

Hardware attestation (TPM/TEE) is required at enterprise scale.

5.4. CB4A-as-a-Service

CB4A's architecture lends itself naturally to a managed service model where a third-party provider operates the broker infrastructure on behalf of customers. In this deployment, the customer runs only the SPIRE agent alongside their AI agents; the PDP, CDP, vault, and audit trail are hosted and operated by the service provider.

This model is particularly relevant because CB4A's functionality overlaps significantly with existing Cloud Access Security Broker (CASB) and Secure Access Service Edge (SASE) platforms. Both CASBs and SASE products already operate inline security mediation layers between users/applications and cloud services. Adding agent credential brokering to these platforms is a natural extension of their existing capability:

* CASB platforms already mediate access between enterprise users and cloud services, enforce DLP policies, and provide audit trails. Adding CB4A's credential brokering extends this mediation to AI agents as a new class of "user."

* SASE platforms already combine network security (firewall, SWG, ZTNA) with WAN optimization. CB4A's PDP/CDP model fits within the SASE policy enforcement architecture as an additional service function.

* Zero Trust Network Access (ZTNA) solutions already enforce per-request access decisions based on identity and context. CB4A's Task Request Envelope is an agent-specific extension of the ZTNA access request model.

Considerations for CB4A-as-a-Service:

* The service provider becomes the trust anchor for credential storage. Customer due diligence on the provider's security posture is critical.

* Latency between customer agents and the hosted broker adds round-trip time to every credential minting operation. Edge deployment of broker components mitigates this.

* Multi-tenancy introduces isolation requirements: one customer's policy engine and credential vault MUST be isolated from another's.

* Regulatory and compliance constraints may require that credentials never leave certain jurisdictions, requiring regional broker deployments.

6. Capability Tiers

CB4A defines three capability tiers. Each builds on the previous one. A home deployment at Tier 1 is a complete system, not an incomplete enterprise deployment.

6.1. Tier 1: Core (Minimum Viable CB4A)

* SPIRE-based workload identity with agent attestation
* PDP with static policy evaluation
* CDP with Model B (short-lived token minting)
* Task Request Envelope schema, validation, and signing
* Tier 1 auto-approval based on policy matching
* Append-only audit trail with structured log entries
* Non-renewable leases on all issued tokens
* Fail-closed degradation for all critical components
* Canary credential seeding in the credential store

6.2. Tier 2: Human Oversight

* Tier 2 HITL approval workflow
* Tier 3 MFA approval (FIDO2/WebAuthn)
* DPoP sender-constrained token binding
* Behavioral baseline collection and anomaly scoring
* Dynamic risk scoring in PDP
* Model A proxy fallback for services without native support
* Approval fatigue countermeasures
* Confused deputy detection
* Emergency break-glass procedure

6.3. Tier 3: Advanced Threat Defense

* Hardware attestation for agent identity (TPM/TEE)
* Runtime behavior monitoring with mid-flight revocation
* Cross-agent correlation engine
* Per-service scope definition refinement
* Log integrity verification via hash chaining
* Adversarial testing program
* Multi-tenant federation via SPIFFE trust domains

7. Security Considerations

The full threat model is provided in Appendix A. Key security considerations include:

* The broker (CDP) is the highest-value target in the architecture. It MUST be hardened with no shell access, restricted network, and credential zeroing after minting (TM-1).
* Ephemeral tokens in agent memory are extractable. DPoP sender-constrained tokens [RFC9449] MUST be used to prevent replay (TM-3).
* Approval fatigue is a human factors risk that cannot be fully solved with technical controls. Challenge approvals and rate limiting reduce but do not eliminate the risk (TM-4).
* Multi-agent scope composition allows individually-scoped credentials to be combined for unauthorized access. Cross- agent correlation is RECOMMENDED (TM-6).
* Fail-closed degradation is critical. CB4A MUST NOT default to fail-open under any failure condition (TM-9).
* Approval channel spoofing is a threat. High-risk approvals MUST use FIDO2/WebAuthn and signed approval responses (TM-10).
* Broker bypass, where agents access services directly without CB4A mediation, MUST be prevented through network-level enforcement, not agent cooperation (TM-11).

8. IANA Considerations

This document makes no requests of IANA.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8693] Jones, M., Nadalin, A., Campbell, B., Ed., Bradley, J., and C. Mortimore, "OAuth 2.0 Token Exchange", RFC 8693, DOI 10.17487/RFC8693, January 2020, <https://www.rfc-editor.org/rfc/rfc8693>.

[RFC9449] Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)", RFC 9449, DOI 10.17487/RFC9449, September 2023, <https://www.rfc-editor.org/rfc/rfc9449>.

9.2. Informative References

[BEYONDCORP] Ward, R. and B. Beyer, "BeyondCorp: A New Approach to Enterprise Security", ;login: vol. 39, no. 6, December 2014.

[LITELLM-SECURITY] "Security Update: Suspected Supply Chain Incident", March 2026, <https://docs.litellm.ai/blog/security-update-march-2026>.

[NIST-ZTA] Rose, S., Borchert, O., Mitchell, S., and S. Connelly, "Zero Trust Architecture", NIST Special Publication 800-207, August 2020, <https://csrc.nist.gov/publications/detail/sp/800-207/final>.

[SPIFFE] "SPIFFE: Secure Production Identity Framework For Everyone", <https://spiffe.io/>.

[SPIRE] "SPIRE: SPIFFE Runtime Environment", <https://github.com/spiffe/spire>.

[TEAMPCP-LITELLM] "TeamPCP Expands Supply Chain Campaign With LiteLLM PyPI Compromise", March 2026, <https://www.infosecurity-magazine.com/news/teampcp-litellm-pypi-supply-chain/>.

[VAULT] "HashiCorp Vault", <https://developer.hashicorp.com/vault>.

Appendix A. Threat Model

This appendix presents a systematic threat analysis of the CB4A architecture. Each threat is identified with a unique ID (TM-N), mapped to the architectural component it targets, assessed for severity, and paired with the mitigation specified elsewhere in this document.

A.1. Methodology

This threat model uses a component-based approach, examining each trust boundary and data flow for potential compromise scenarios. Threats are classified by target, attacker model, severity (Critical/High/Medium), and mitigation cross-reference.

A.2. TM-1: Broker Compromise

Target: Credential Delivery Point (CDP). Severity: CRITICAL.

The CDP must access real credentials to mint derivative tokens. An attacker who compromises the CDP process (via RCE, supply chain attack, or SSRF) can intercept credentials during the minting operation.

Mitigation: CDP never caches credentials beyond minting; credentials zeroed immediately. CDP runs in hardened runtime. Audit trail on separate trust boundary. PDP/CDP separation prevents approval of arbitrary requests.

Residual Risk: During the brief minting window, the credential exists in CDP memory. This is irreducible for Model B.

A.3. TM-2: Revocation Propagation Failure

Target: Token lifecycle management. Severity: HIGH.

Immediate revocation across independent downstream services violates distributed systems fundamentals. Services cache auth decisions and experience network partitions.

Mitigation: Three configurable revocation strategies. Ultra-short TTLs as primary control. CB4A proxy for unreliable services. Non-renewable leases.

Residual Risk: A propagation window will always exist for Model B tokens against caching services.

A.4. TM-3: Token Theft and Replay

Target: Agent runtime memory. Severity: HIGH.

Short-lived tokens in agent memory can be extracted and replayed from an external system within the TTL window.

Mitigation: DPoP sender-constrained tokens [RFC9449] bind each token to an ephemeral key pair. Stolen tokens are cryptographically useless without the private key.

Residual Risk: Simultaneous extraction of token AND DPoP private key defeats the binding.

A.5. TM-4: Approval Bypass and Fatigue

Target: Tiered approval service. Severity: HIGH.

Bypass: Compromised agent crafts requests to route through auto-approval. Fatigue: High-volume legitimate requests condition approvers to rubber-stamp.

Mitigation: Rate limiting, challenge approvals, cool-downs. Cross-agent correlation detects chaining. FIDO2 at Tier 3.

Residual Risk: Human factors cannot be fully solved with technical controls.

A.6. TM-5: Justification Field Gaming

Target: Task Request Envelope. Severity: MEDIUM.

Compromised agent crafts adversarially-optimized justification text.

Mitigation: Justification is NOT an authorization input. Attack surface removed by design.

Residual Risk: None if implementations adhere to specification.

A.7. TM-6: Multi-Agent Scope Composition

Target: Blast radius containment. Severity: HIGH.

An attacker compromising multiple agents can compose individually-scoped credentials for unauthorized access.

Mitigation: Cross-agent correlation engine. Behavioral baselines. Canary credentials. Short TTLs.

Residual Risk: Learning period for new deployments.

A.8. TM-7: Audit Log Compromise

Target: Immutable audit trail. Severity: HIGH.

Attacker tampers with or suppresses audit log entries.

Mitigation: Separate trust boundary. Hash chaining. Fail-closed on log pipeline failure.

Residual Risk: Dual compromise of broker AND audit trail.

A.9. TM-8: Policy Engine Injection

Target: Policy Decision Point. Severity: MEDIUM-HIGH.

Crafted envelope values exploit policy evaluation via injection.

Mitigation: Typed, compiled policy language. Input validation. No string interpolation.

Residual Risk: Low with correct implementation.

A.10. TM-9: Fail-Open Under Pressure

Target: Secure degradation model. Severity: HIGH.

Organizational pressure creates undocumented bypass mechanisms.

Mitigation: Explicit fail-closed for every failure mode. Dual-key break-glass with mandatory post-review.

Residual Risk: Governance and compliance issue.

A.11. TM-10: Approver Spoofing

Target: Tiered approval service, approval routing. Severity: HIGH.

Attacker impersonates an authorized approver via compromised communication channel or intercepted approval webhook.

Mitigation: FIDO2/WebAuthn for Tier 3. Cryptographically signed approval responses. Out-of-band verification on authenticated dashboard. Approval routing to agent owners.

Residual Risk: Physical theft of approver’s security key.

A.12. TM-11: Broker Bypass

Target: CB4A mediation layer. Severity: HIGH.

A compromised agent attempts to access target services directly, bypassing CB4A entirely. If the agent can reach the target API without going through the broker, all CB4A controls (approval, scoping, audit, TTL) are circumvented. This is the credential broker equivalent of the bypass problem that Cloud Access Security Brokers (CASBs) have addressed for over a decade.

Attack Scenario: 1. Compromised agent discovers that network-level controls do not block direct access to a target API endpoint. 2. Agent uses credentials obtained through a prior legitimate CB4A interaction (within TTL) to call the service directly, outside the audit trail. 3. Alternatively: agent accesses credentials from an environment variable, configuration file, or process memory of another application on the same host.

Mitigation: Network-level enforcement blocking agent direct access to brokered services (Section 4.10). Vault access restricted to CDP only. Service-level audit log monitoring for calls not originating from CB4A. DNS-level controls resolving brokered endpoints to CB4A proxy.

Residual Risk: Agents running on hosts with broad network access and weak segmentation. Environments where network policy enforcement is not feasible (e.g., shared development machines without container isolation).

A.13. Threat Summary

ID	Threat	Severity	Primary Mitigation
TM-1	Broker compromise	CRITICAL	Hardened runtime, credential zeroing
TM-2	Revocation failure	HIGH	Ultra-short TTLs, configurable strategies
TM-3	Token theft/ replay	HIGH	DPoP sender-constrained tokens
TM-4	Approval bypass/ fatigue	HIGH	Rate limiting, challenge approvals
TM-5	Justification gaming	MEDIUM	Field excluded from authorization
TM-6	Multi-agent composition	HIGH	Cross-agent correlation engine
TM-7	Audit log compromise	HIGH	Separate trust boundary, hash chaining
TM-8	Policy engine injection	MED-HIGH	Typed policy language
TM-9	Fail-open pressure	HIGH	Explicit fail-closed, break-glass
TM-10	Approver spoofing	HIGH	FIDO2/WebAuthn, signed approvals
TM-11	Broker bypass	HIGH	Network enforcement, vault isolation

Table 7

Acknowledgments

This specification was developed with input from multi-agent architectural review and adversarial analysis.

Author’s Address

Kenneth G. Hartman
SANS Institute
Email: khartman@sans.org