AGENT SECURITY & IDENTITY ARCHITECTURE • FIELD GUIDE

Agent Auth, Identity & Scoped Permissions

Secure multi-tenant AI agents and MCP servers in production using OAuth 2.1 + PKCE, short-lived token brokering, least-privilege RBAC/ABAC scopes, human approval gates, and tamper-evident audit logs.

OAuth 2.1 & PKCE | RFC 9728 & CIMD | Token Vaults & OBO | Least-Privilege Scopes | Confused Deputy Defense | Step-Up & Audit Trails



Zero Raw Secrets in Context

Keep OAuth refresh tokens and API keys inside server-side Token Vaults. Agents receive only short-lived, audience-bound credentials per tool call.

Dual-Identity Delegation

Bind every action to both the human principal (`sub`) and the acting agent (`act`) via RFC 8693 Token Exchange and strict RFC 8707 audience checks.

Blast-Radius Containment

Combine RBAC/ABAC scope ceilings, out-of-band human step-up approvals, Postgres RLS, and instant kill switches for SOC 2 & EU AI Act readiness.

Why Agent Auth Breaks Traditional API Keys

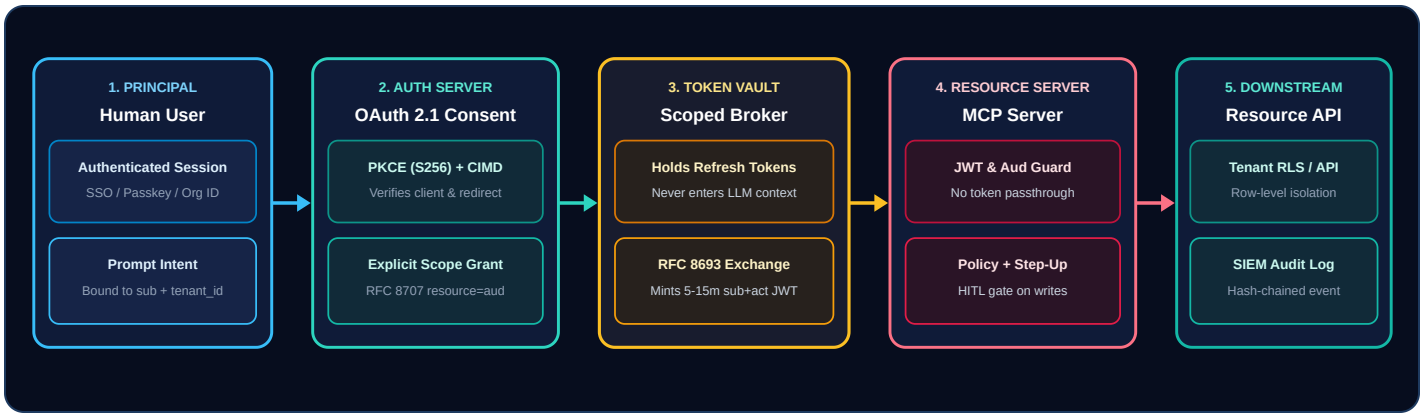
Traditional SaaS apps call APIs with deterministic code paths. Autonomous AI agents choose which tools to invoke at runtime, traverse multi-tenant data, and process untrusted external text. Handing an agent a static service key turns a single prompt injection into a full-tenant breach.

Quick Navigation

1. Mental Model: Zero-Trust Token Flow p. 2
2. Auth Pattern Decision Matrix p. 3
3. Annotated Identity & Consent HUD p. 4
4. MCP OAuth 2.1 & RFC 9728 Architecture p. 5
5. 5-Part Least-Privilege Policy Anatomy p. 6
6. Token Brokering & Managed Vaults p. 7
7. Defeating Confused Deputy & Leaks p. 8
8. 5 Copy-Paste Auth & MCP Recipes p. 9
9. Governance, Audit Trails & Kill Switches p. 10
10. Cross-Tool Enterprise Auth Pipeline p. 11
11. Troubleshooting Clinic & Pre-Flight p. 12
Core Rule: Scoped, Short-Lived, Auditable All

The Production Agent Authorization Loop

In late-2026 production architectures, the LLM never touches credentials. Instead, an authenticated user grants narrow consent via OAuth 2.1, a server-side broker mints ephemeral audience-bound tokens, and every MCP server independently validates identity, scope, and tenant isolation before touching downstream APIs:



1 Identity != Authorization

Knowing *who* the user is (OIDC) is only step one. Every tool call must check *what* this specific agent is allowed to do on *which* tenant record right now.

2 Ephemeral by Default

Cap agent access tokens at **5–15 minutes**. Short TTLs neutralize stolen tokens and make policy changes or offboarding propagate almost instantly.

3 Trust Boundaries Hold

Never accept `user_id` or `tenant_id` as LLM tool arguments. Derive identity strictly from cryptographically verified server-side JWT claims.

The 2026 Industry Consensus: Governance at the Tool-Calling Layer

Following MCP's move to the Linux Foundation's Agentic AI Foundation and the OWASP MCP Top 10, production security teams treat the **MCP server boundary**—not the prompt—as the primary enforcement point. Prompt instructions are advisory; OAuth 2.1 tokens, RBAC/ABAC policies, and database RLS are binding.

Choosing the Right Agent Auth Architecture

No single OAuth flow fits every agent workload. Interactive chat copilots, multi-hop enterprise orchestrators, third-party SaaS connectors, and headless cron agents each demand a distinct credential pattern.

Auth Pattern	Verdict	Best-Fit Workload	Identity & Token Mechanics	Primary Risk / Guardrail
Static API Keys (Shared Env Secret)	DON'T	Local single-user `stdio` dev prototypes only. Never multi-tenant prod.	Long-lived bearer string with broad org-wide admin rights; zero user attribution.	Total blast radius on leak; impossible per-user revocation or SOC 2 audit trail.
OAuth 2.1 + PKCE (Auth Code Flow)	DEFAULT	Interactive agents & remote MCP servers acting for a live signed-in user.	RFC 9728 discovery + S256 PKCE + RFC 8707 `resource` binding to MCP server `aud`.	Must enforce exact `redirect_uri` & per-client consent to block confused deputy.
OBO Delegation (RFC 8693 Exchange)	ENTERPRISE	Multi-hop agent → MCP server → internal microservices or sub-agents.	Exchanges upstream token for a narrower downstream JWT carrying `sub` (user) + `act` (agent).	Prevents token passthrough; enforce strict scope attenuation at every hop.
Token Vaults (Managed Broker)	SAAS TOOLS	Agents calling 3rd-party APIs (Gmail, Slack, GitHub, Linear, Salesforce).	Vault stores encrypted refresh tokens & injects fresh short-lived access tokens at runtime.	Never return raw vault tokens to LLM; bind lookup strictly to verified session `sub`.
Workload OIDC (SPIFFE / M2M)	HEADLESS	Autonomous background agents (CI triage, nightly billing reconciliation).	Short-lived cryptographic workload JWT (AWS/GCP/Vercel OIDC or SPIFFE SVID), no user present.	Zero human session—require strict ABAC resource filters, spend caps, & circuit breakers.

Fast Decision Heuristic: 4 Questions to Pick Your Stack

Q1 Is a human user present at invocation?

Yes → Use **OAuth 2.1 + PKCE** so permissions mirror the user's real RBAC role.
No (scheduled/webhook) → Use **Service Workload OIDC** with a dedicated non-human principal (e.g., `agent:nightly-triage`) and narrow read/write ceilings.

Q2 Does the agent call 3rd-party SaaS APIs?

Yes → Never hand-roll encrypted refresh token tables. Use a managed **Token Vault** (Auth0 Token Vault, WorkOS, Vercel Connect) to handle OAuth consent, rotation, and revocation.

Q3 Does an MCP server call other internal APIs?

Yes → Do *not* forward the client's bearer token. Perform **RFC 8693 On-Behalf-Of Token Exchange** to mint a downstream JWT scoped strictly to the target microservice.

Q4 Can the tool mutate money, data, or prod?

Yes → Standing OAuth tokens are insufficient. Layer **Step-Up Human Approval** (CIBA push, Slack/WebAuthn confirm) bound to the exact tool arguments hash.

Real-World Hybrid Pattern: Combine OAuth 2.1 Edge + Vault Downstream

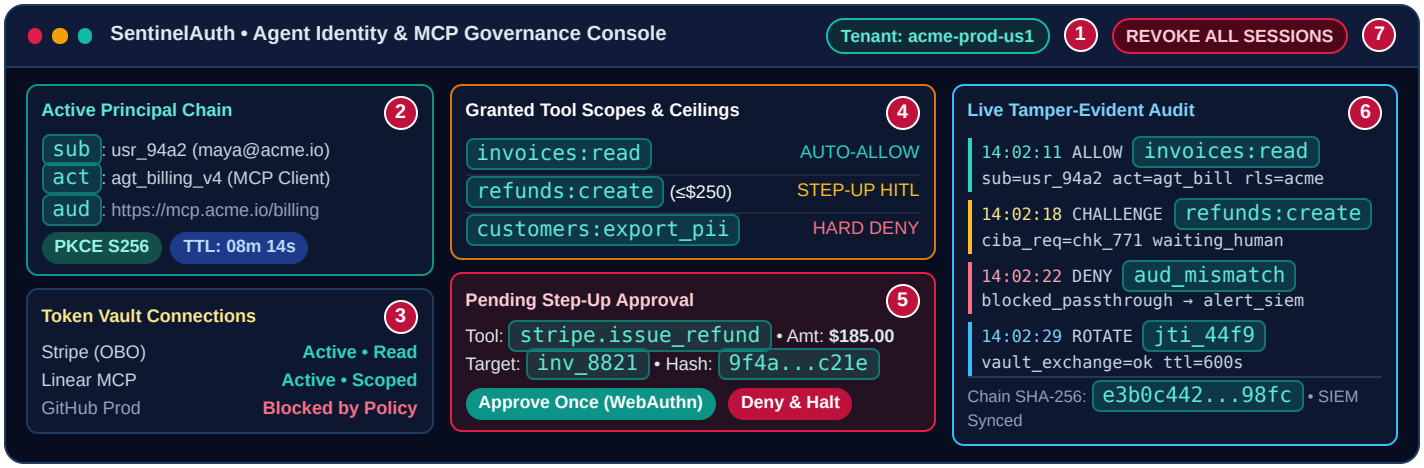
Most production AI apps use **two layers**: (1) Your app authenticates the user via OAuth 2.1 / OIDC to your remote MCP server, and (2) when a tool needs Slack or Linear, the MCP server asks a **Token Vault** for a 5-minute user-scoped downstream token. The LLM never sees either credential.

Transport Rule from the MCP Specification (2026)

Streamable HTTP (Remote MCP): MUST implement OAuth 2.1 + RFC 9728 Protected Resource Metadata. • **Local `stdio` MCP:** SHOULD NOT implement OAuth flows; read user-local credentials from environment variables or OS keychain instead.

Inside an Agent Identity & Scope Consent Console

Production agent platforms expose a dedicated security control plane where admins and end-users inspect active agent identities, narrow OAuth scopes, approve step-up requests, and pull instant kill switches.



7 Controls Every Production Agent Security Plane Needs

1 Tenant Isolation Context Pill

Pins every policy view, token exchange, and audit filter to a verified `tenant_id` so operators and agents never cross workspace boundaries.

2 Dual Principal Chain (`sub` + `act`)

Displays both the delegating human and the specific agent version, plus RFC 8707 audience lock and remaining token TTL countdown.

3 Managed Token Vault Connectors

Shows which third-party OAuth connections are active, read-only, or blocked—without ever exposing raw access or refresh secrets.

4 Scope Ceiling & Policy Matrix

Classifies every MCP tool into **Auto-Allow** (safe reads), **Step-Up HITL** (bounded writes), or **Hard Deny** (PII export / destructive ops).

5 Argument-Bound Step-Up Approval

Locks human approval to a cryptographic hash of the exact tool arguments so a compromised agent cannot swap `$185` for `$18,500` post-approval.

6 Live Tamper-Evident Audit Stream

Streams hash-chained allow/deny/challenge events directly to Datadog, Splunk, or SIEM with full `sub`, `act`, and `jti` correlation.

7 Emergency Kill Switch (Revoke All Sessions)

A single click must invalidate active refresh tokens, push revoked `jti` claims to edge caches, and trip the MCP gateway circuit breaker in under 2 seconds—halting runaway agent loops before damage compounds.

How Remote MCP Servers Enforce OAuth 2.1

The Model Context Protocol standardizes remote HTTP authentication on an **OAuth 2.1 profile**. The MCP server acts strictly as an *OAuth Resource Server*: it never handles passwords, never runs login UIs, and delegates token issuance to an external Authorization Server (AS).



Figure 5.1 — Cryptographic Handshake Boundary: Discovery via `RFC 9728`, PKCE + audience-bound issuance at the Authorization Server, and strict token validation at each isolated MCP Resource Server.

The 5-Step MCP Authorization Handshake (2026 Spec)

Step	RFC / Mechanism	Exact Production Behavior
1	**401 Challenge + RFC 9728**	Client calls `POST /mcp` without a token. Server replies `401 Unauthorized` with `WWW-Authenticate: Bearer resource_metadata="https://mcp.example.com/.well-known/oauth-protected-resource"`.
2	**Metadata Discovery** (RFC 9728 & RFC 8414)	Client fetches Protected Resource Metadata to locate `authorization_servers` and supported scopes, then fetches AS metadata (`/.well-known/oauth-authorization-server`) to confirm `S256` PKCE support.
3	**Client Identity** (Pre-Reg, CIMD, or DCR)	Client identifies itself using pre-registration (internal apps), a **Client ID Metadata Document (CIMD)** hosted at an HTTPS URL matching `client_id`, or RFC 7591 Dynamic Client Registration (DCR).
4	**PKCE + RFC 8707** Resource Indicator	User completes browser consent. Client sends `code_challenge` (S256) and `resource=https://mcp.example.com/mcp` so the AS locks the issued JWT's `aud` claim exclusively to that MCP server.
5	**Stateless JWT Verification**	MCP server verifies JWT signature via AS JWKS, checks `iss`, `exp`, exact `aud` match, and required tool `scope` before executing any `tools/call` handler.

CIMD vs. DCR: Why 2026 Favors CIMD

Dynamic Client Registration (DCR) can pollute your IdP with thousands of anonymous client rows. **CIMD** lets an MCP client host a static JSON file at `https://app.co/oauth/client-metadata.json` as its `client_id`. Always guard CIMD fetches against SSRF and pin `redirect_uris`.

Mandatory Security Invariants

- **PKCE S256 only**—reject `plain` and un-PKCE'd flows.
- **Strict `aud` match**—reject tokens minted for other APIs.
- **No wildcard redirects**—exact scheme, host, and path.

The 5-Part Agent Authorization Policy Anatomy

Coarse OAuth scopes like `crm:admin` or `read_write` give an agent keys to the kingdom. Production agent security pairs narrow OAuth scopes with a **5-part runtime policy** evaluated on every tool invocation.

COLOR-CODED RUNTIME POLICY FORMULA

Allow **1. Principal (sub=usr_94a2, act=agt_billing_v4, tenant=acme)** to execute **2. Action (stripe.refund.create)** against **3. Resource (invoices where org_id == token.tenant_id AND status == 'paid')** subject to **4. Scope Ceiling (max $250/tx, 3/hr, TTL ≤ 10m)** and require **5. Step-Up Trigger (human WebAuthn/CIBA approval if amount > $50)**.

1. PRINCIPAL **Dual Identity**

Bind policy to *both* the human (`sub`, org role) and the agent (`act`, model tier, client ID). A user's admin role should **never** automatically grant admin writes to an untrusted third-party agent.

2. ACTION **Verb-Level Tool Split**

Split MCP tools by risk class: `tickets:read`, `tickets:draft_reply`, and `tickets:send_external`. Never bundle read and destructive write capabilities inside one multipurpose tool.

3. RESOURCE **Tenant & Record Filter**

Enforce object-level checks (ABAC/ReBAC): the invoice, repo, or folder must belong to `jwt.tenant_id` and the user's explicit ACL group—verified server-side, never trusted from prompt text.

4. SCOPE CEILING **Hard Blast Radius**

Cap quantitative impact in code: maximum dollar amount, row count (`LIMIT 50`), rate limit per minute, and token lifetime (`exp - iat ≤ 900s`). Ceilings stop prompt-injected loops cold.

5. STEP-UP TRIGGER **Out-of-Band Human Approval Gate**

Any irreversible action—funds transfer, database deletion, permission grant, or external email/webhook—pauses execution and fires a signed approval challenge (WorkOS Step-Up, Auth0 CIBA, or Slack interactive card) bound to a SHA-256 hash of the exact tool parameters.

RBAC vs. ABAC vs. ReBAC for Multi-Tenant Agents

Model	What It Evaluates	Best Agent Use Case	Recommended Engine
RBAC	Static org roles (`viewer`, `support`, `admin`) in JWT claims.	Tool allowlists & coarse read vs. write tiering per agent persona.	IdP claims (Okta, WorkOS, Clerk, Auth0).
ABAC	Runtime attributes: `amount`, `env=prod`, `data_class=pii`, time.	Dynamic scope ceilings, spend limits, & step-up triggers on tool args.	OPA (Rego), AWS Cedar, or TS policy guard.
ReBAC	Fine-grained object graph (User → Folder → Doc).	RAG & MCP search across Google Drive, Notion, or multi-team repos.	OpenFGA, AuthZed (SpiceDB), WorkOS FGA.

Token Vaults: Keeping Raw Credentials Out of LLM Context

The fastest way to leak customer OAuth tokens is passing them into agent state, tool outputs, or prompt memory. A **Token Vault** decouples credential storage from agent reasoning—storing encrypted refresh tokens centrally and injecting ephemeral access tokens only at the HTTP execution boundary.

Late-2026 Agent Token Vault & Auth Platform Landscape

Platform	Core Agent Capability	MCP & Delegation Support	Best Fit
Auth0 for AI Agents (Token Vault)	Encrypted 3rd-party OAuth storage (Google, Slack, GitHub), CIBA Async Auth, Agent as Principal.	GA Auth for MCP (OAuth 2.1 + CIMD + RFC 8693 OBO token exchange).	B2C/B2B SaaS needing turnkey 3rd-party vaults & async push approvals.
WorkOS (AuthKit & Agent Vault)	Enterprise SSO/SCIM, Step-Up Auth for agents, Fine-Grained Auth (FGA), & Agent Vault.	Full OAuth 2.1 Authorization Server for remote MCP + signed-in browser sessions.	B2B enterprise apps selling into Okta/Entra ID customers with strict IT controls.
Clerk	Drop-in OAuth 2.1 consent UI, Org RBAC, session-bound JWTs, & edge verification SDKs.	Native MCP OAuth provider & `verifyToken` middleware for Next.js/Workers.	Product-led SaaS teams already on Clerk wanting rapid remote MCP auth.
Stytch (Connected Apps)	Drop-in OAuth 2.1 + Trusted Auth Tokens (layers onto any existing IdP/CIAM stack).	Full DCR & CIMD support, org-level client allowlists, & IT admin deduplication.	Teams with legacy/custom auth who want MCP OAuth without re-platforming.
Vercel Connect (`@vercel/connect`)	Managed OAuth/API credential broker for AI SDK; automatic OIDC & proactive refresh.	`connectAuthProvider` for `@ai-sdk/mcp` with `ConsentRequiredError` flow.	Next.js / Vercel AI SDK agents calling Linear, GitHub, & remote MCP servers.

4-Step Runtime Token Brokering Lifecycle

1 Derive Subject from Verified Session

Extract `sub` and `tenant_id` from the authenticated HTTP session cookie or bearer JWT. Never read a user ID from the chat payload or LLM tool arguments.

2 Just-in-Time Vault Lookup

Inside the server-side tool executor (not the LLM), request a token for `(sub, provider, required_scopes)`. If missing or revoked, throw a `ConsentRequiredError` and render an in-chat OAuth connect card.

3 Proactive Refresh & Scope Narrowing

The vault transparently rotates expiring tokens using its KMS-encrypted refresh token and returns a minimal, short-lived access token scoped only to the requested tool action.

4 Execute, Scrub & Discard

Inject the token into the outbound HTTPS `Authorization` header, execute the API call, scrub any auth headers or cookies from error traces, and discard the token from memory immediately.

Critical Rule: Never Hold a Streaming Request Open During OAuth Consent

When a Token Vault signals missing user consent (e.g., Vercel Connect's `ConsentRequiredError`), terminate the current turn cleanly and return a structured consent URL to the UI. Once the user completes OAuth in the browser, start a fresh authenticated turn to resume the tool call.

Vault Storage Hygiene (If Self-Hosting)

Envelope-encrypt every refresh token with per-tenant KMS data keys (AES-256-GCM), enforce token rotation with reuse detection, log every vault read to SIEM, and never serialize credential objects into LangGraph/workflow checkpoints or Redis caches.

Defeating Confused Deputy, Passthrough & Cross-Tenant Leaks

The MCP Authorization Security Considerations highlight four high-severity failure modes that break multi-tenant isolation in real deployments. Every production MCP server must close all four.

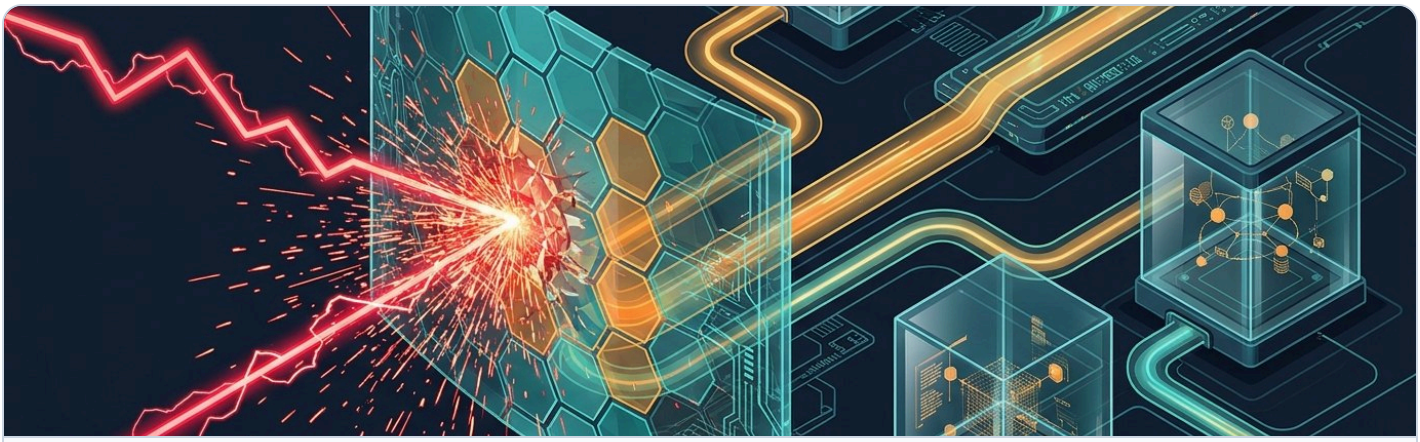


Figure 8.1 — Zero-Trust Perimeter Enforcement: Audience-bound validation, per-client consent binding, and database-level tenant isolation shatter confused deputy and cross-tenant exploits before they reach customer vaults.

1. Confused Deputy on MCP Proxy

Attack: An MCP proxy uses a single static `client_id` to talk to an upstream IdP. A malicious agent crafts an auth link; because the victim already consented once, the upstream IdP silently issues a code to the attacker's redirect URI.

Defense: Require explicit **per-client consent** on the proxy itself (bound via HttpOnly signed consent cookie + PKCE + `state`) before redirecting upstream, and forbid wildcard `redirect_uris`.

2. Token Passthrough Anti-Pattern

Attack: An MCP server blindly forwards the client's bearer JWT to an upstream API. This bypasses downstream audience checks, breaks audit attribution, and lets a compromised tool replay the user's broad token elsewhere.

Defense: Strictly forbidden by the MCP spec. Validate `aud == my_mcp_url` (RFC 8707) and use **RFC 8693 OBO Token Exchange** or a Token Vault to mint a separate downstream token.

3. Cross-Tenant Context & Cache Leak

Attack: A shared MCP connection pool, prompt cache, or vector index forgets to filter by tenant, or accepts `tenant_id` from an LLM tool argument—allowing prompt injection to query Tenant B's records.

Defense: Strip `tenant_id` from tool schemas. Inject `SET LOCAL app.tenant_id = jwt.tenant_id` inside every Postgres transaction (RLS) and partition vector namespaces per tenant.

4. Tool Poisoning & Shadow Escalation

Attack: Hidden instructions inside a 3rd-party MCP tool description or retrieved document coerce the LLM into invoking a high-privilege tool (e.g., reading SSH keys or exfiltrating CRM rows via webhook).

Defense: Enforce a static **per-agent tool allowlist** at the gateway, pin tool schema hashes against rug-pulls, and require out-of-band Step-Up approval for any external egress or write.

Golden Defense-in-Depth Rule: Assume the LLM Is Compromisable

Treat every tool call emitted by the model as **untrusted user input**. If an attacker takes full control of the LLM's reasoning via indirect prompt injection, your OAuth 2.1 audience lock, RBAC/ABAC scope ceiling, Step-Up gate, and Postgres RLS must still prevent unauthorized data access or mutation.

Production Auth & MCP Middleware Snippets

Drop these five battle-tested patterns into your TypeScript/Node.js MCP servers and Next.js agent routes to meet late-2026 OAuth 2.1, audience binding, and human-in-the-loop standards.

#	Recipe	What It Solves	Key Standard / Primitive
1	**RFC 9728 PRM Endpoint**	Zero-config OAuth discovery & `401` challenge.	`/.well-known/oauth-protected-resource`
2	**JWT + Audience Guard**	Verifies `iss`, `aud`, `exp`, & tool `scope`.	`jose` `jwtVerify()` + RFC 8707
3	**Step-Up Approval Gate**	Pauses writes until human signs arg-hash.	SHA-256 arg digest + CIBA / WebAuthn
4	**OBO Token Exchange**	Replaces passthrough with scoped downstream JWT.	RFC 8693 `grant-type:token-exchange`
5	**Token Vault Boundary**	Fetches SaaS tokens & handles consent errors.	`@vercel/connect` / Auth0 Token Vault

Recipe 1: RFC 9728 Protected Resource Metadata & 401 Challenge

```
// GET /.well-known/oauth-protected-resource  (Content-Type: application/json)
{ "resource": "https://mcp.acme.io/billing", "authorization_servers": ["https://auth.acme.io"],
  "scopes_supported": ["invoices:read", "refunds:create"], "bearer_methods_supported": ["header"] }
// On missing/invalid token, return HTTP 401 with:
WWW-Authenticate: Bearer resource_metadata="https://mcp.acme.io/.well-known/oauth-protected-resource"
```

Recipe 2: MCP Server JWT, Audience (`aud`) & Scope Verification

```
import { createRemoteJWKSet, jwtVerify } from "jose";
const JWKS = createRemoteJWKSet(new URL("https://auth.acme.io/.well-known/jwks.json"));
export async function verifyMcpToken(authHeader: string | undefined, requiredScope: string) {
  const token = authHeader?.replace(/^Bearer\s+/i, "");
  if (!token) throw new Error("MISSING_BEARER_TOKEN");
  const { payload } = await jwtVerify(token, JWKS, {
    issuer: "https://auth.acme.io", audience: "https://mcp.acme.io/billing", clockTolerance: 5,
  });
  if (!String(payload.scope ?? "").split(" ").includes(requiredScope)) throw new Error("INSUFFICIENT_SCOPE");
  return { sub: String(payload.sub), act: (payload.act as any)?.sub, tenantId: String(payload.tenant_id) };
}
```

Recipe 3: Argument-Bound Step-Up Human Approval Middleware

```
import { createHash } from "crypto";
export async function requireStepUpApproval(ctx: { sub: string; tenantId: string }, tool: string, args: unknown) {
  const argsHash = createHash("sha256").update(JSON.stringify(args)).digest("hex");
  const apr = await approvalsDb.findValid({ sub: ctx.sub, tenantId: ctx.tenantId, tool, argsHash });
  if (apr && !apr.consumed) { await approvalsDb.consumeOnce(apr.id); return apr.id; }
  const ch = await approvalsDb.createChallenge({ ...ctx, tool, argsHash, ttlSec: 300 });
  return { status: "APPROVAL_REQUIRED", approvalUrl: ch.url, expiresIn: 300 };
}
```

Pro Tip for Recipes 4 & 5 (OBO + Token Vaults)

When calling downstream APIs from an MCP handler, pass `grant_type=urn:ietf:params:oauth:grant-type:token-exchange` with `subject_token` + target `resource`, or call `connectAuthProvider` / Auth0 Token Vault with verified `ctx.sub`. Never forward the inbound `Authorization` header.

Tamper-Evident Audit Logs & Instant Kill Switches

When an enterprise CISO asks, *"Which human authorized this agent to issue a refund at 2:14 AM, and how do we stop it in 2 seconds?"*, your audit schema and revocation architecture must answer with cryptographic certainty.

Structured Dual-Identity Audit Event (Hash-Chained JSON)

```
{
  "event_id": "evt_01JCY9M4K2...",          "timestamp": "2026-11-22T14:02:18.412Z",
  "tenant_id": "acme-prod-us1",             "trace_id": "trc_88f1a2c90b",
  "principal_sub": "usr_94a2",             "actor_agent_id": "agt_billing_v4",
  "token_jti": "jti_44f9c120",             "mcp_tool": "stripe.issue_refund",
  "args_sha256": "9f4a8b12...c21e",        "policy_decision": "ALLOW_WITH_STEP_UP",
  "approval_id": "apr_7710_webauthn",      "prev_event_sha256": "e3b0c442...98fc"
}
```

T1 Tool Circuit Breaker
Disable a single compromised or noisy MCP tool globally via feature flag in <1s without disrupting read-only agent workflows.

T2 Session & JTI Revoke
Revoke a user's refresh token at the IdP + push active `jti` into a Redis/edge denylist so short-lived JWTs die immediately.

T3 Tenant Emergency Freeze
One-click org-level kill switch halts all autonomous agent traffic and suspends Token Vault exchanges during an incident.

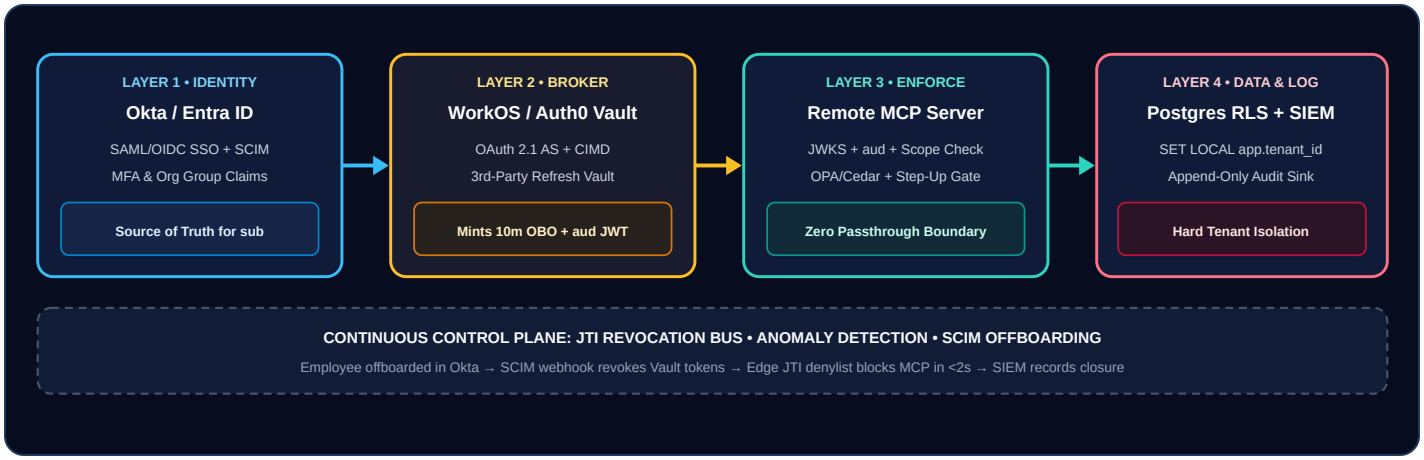
Compliance Mapping: SOC 2, ISO 27001, NIST & EU AI Act (2026)

Framework	Enterprise Auditor Requirement	How This Agent Auth Architecture Satisfies It
SOC 2 Type II (CC6.1 / CC6.3)	Logical access controls, least-privilege provisioning, & timely offboarding.	SCIM/SSO deprovisioning immediately invalidates OAuth refresh tokens; 10-min JWT TTL caps residual window.
ISO/IEC 27001 (A.5.15 / A.8.15)	Unique actor attribution & tamper-protected security event logging.	Every log carries `sub` + `act` + `args_sha256`, streamed to WORM/append-only SIEM storage with hash chaining.
EU AI Act (Art. 12 & 14)	Automatic event logging & effective human oversight for high-risk actions.	Mandatory Step-Up HITL approval gates on financial/legal mutations + immutable approval receipts linked to `trace_id`.
NIST AI Agent Standards (2026)	Verifiable non-human identity, delegation chains, & bounded autonomy.	RFC 8693 delegation tokens + explicit 5-part scope ceilings + centralized MCP gateway inventory.

PII & Secret Redaction Before SIEM Ingestion
Never log raw tool payloads or bearer tokens into audit logs. Store a **SHA-256 hash** of the canonicalized arguments (`args_sha256`) alongside safe metadata (`amount`, `record_id`, `tool_name`) so auditors can prove integrity without turning your SIEM into a PII honey pot.

End-to-End Multi-Tenant Agent Security Pipeline

Real production reliability comes from wiring four specialized layers together so no single layer bears the full burden of security: Enterprise IdP → Token Vault → Remote MCP Gateway → Postgres RLS + SIEM.



Stage-by-Stage Latency & Security Budget

Pipeline Stage	Added Latency	Cryptographic / Policy Guarantee	Failure Mode Behavior (Fail-Closed)
1. IdP SSO & SCIM	0ms (cached)	Enterprise MFA & authoritative org membership.	SCIM delete event immediately suspends all agent grants.
2. Token Vault / AS	15–45ms (cached)	KMS-encrypted refresh storage; RFC 8707 `aud` lock.	Returns `ConsentRequiredError`; never falls back to shared key.
3. MCP Gateway	3–8ms (local JWKS)	Stateless JWT verify + ABAC ceiling + tool allowlist.	Returns `401/403` with RFC 9728 metadata; logs deny event.
4. Postgres RLS + SIEM	<2ms (index)	Kernel-level row filtering even if app SQL has a bug.	Missing `app.tenant_id` session var returns 0 rows (default deny).

Postgres RLS Snippet for MCP Handlers

```
CREATE POLICY tenant_isolation ON invoices
  USING (tenant_id = current_setting('app.tenant_id',
true));
-- Inside every verified MCP DB transaction:
SET LOCAL app.tenant_id = $1; -- from verified JWT
only
```

Why This Stack Wins Enterprise Security Reviews

You add under **55ms** of auth overhead per tool call while eliminating the top 4 reasons enterprise agent pilots stall in InfoSec review: shared credentials, missing user attribution, cross-tenant SQL risk, and unrevokable background sessions.

Troubleshooting Clinic (Top 7 Fixes) & Pre-Flight

Most MCP and agent auth outages stem from seven predictable misconfigurations. Bookmark this clinic to diagnose `401` loops, audience rejections, and consent failures in minutes.

Symptom	Root Cause	Exact Production Fix
1. Infinite 401 Discovery Loop	Missing `resource_metadata` param in `WWW-Authenticate` header or CORS blocked.	Return exact RFC 9728 URL in `401` header and enable `Access-Control-Allow-Origin` on `/.well-known/*`.
2. `aud` Claim Mismatch	Trailing slash mismatch (`/mcp` vs `/mcp/`) or client omitted RFC 8707 `resource`.	Canonicalize the MCP server URI identically across PRM metadata, AS audience config, and `jwtVerify()`.
3. CIMD Fetch Rejected	`client_id` URL lacks a path component or fails AS SSRF/private-IP blocklist.	Host CIMD JSON at a public HTTPS path (e.g., `/oauth/client-metadata.json`) matching `client_id` verbatim.
4. Mid-Run 401 After Revoke	User revoked 3rd-party OAuth in Slack/Vercel while agent held a cached token.	Catch upstream `401` at tool boundary, evict vault cache, and return a clean re-consent link to the user.
5. Step-Up Approval Replay	Approval token wasn't bound to arguments or wasn't marked single-use.	Hash canonical JSON args (`args_sha256`) into the approval record and atomically consume on first execution.
6. Cross-Tenant Empty / Leak	Connection pool reused a previous request's Postgres `SET` session variable.	Always use `SET LOCAL app.tenant_id` inside an explicit `BEGIN ... COMMIT` transaction block.
7. Raw Token in LLM Trace	Tool error serializer dumped full Axios/fetch request headers into tool output.	Strip `Authorization`, `Cookie`, and `x-api-key` headers in an outbound error interceptor before returning to LLM.

6-Point Production Pre-Flight Checklist

- ☑ **OAuth 2.1 + RFC 9728 Live:** Remote MCP returns valid PRM JSON, enforces `S256` PKCE, and rejects any JWT without exact `aud` match.
- ☑ **Zero Token Passthrough:** Verified that inbound client JWTs are never forwarded to upstream APIs; OBO exchange or Token Vault is used instead.
- ☑ **Vault & Context Isolation:** Confirmed zero refresh tokens or API keys appear in LLM prompts, tool responses, or workflow checkpoints.
- ☑ **Scope Ceilings & Step-Up:** Every mutating tool has hard rate/dollar caps and triggers argument-bound human approval above threshold.
- ☑ **Multi-Tenant RLS Verified:** Integration test proves an authenticated Tenant A agent receives 0 rows when prompted to fetch Tenant B IDs.
- ☑ **Kill Switch & SIEM Tested:** Simulated session revocation halts active agent tool calls in <2s and writes a hash-chained `sub+act` audit event.

The Golden Rule of Production Agent Identity

Give every agent the **smallest scope**, for the **shortest lifetime**, bound to a **verifiable human + agent identity chain**—and assume the prompt layer can always be tricked.

READY FOR PROD