Privedge

TECHNICAL ARCHITECTURE

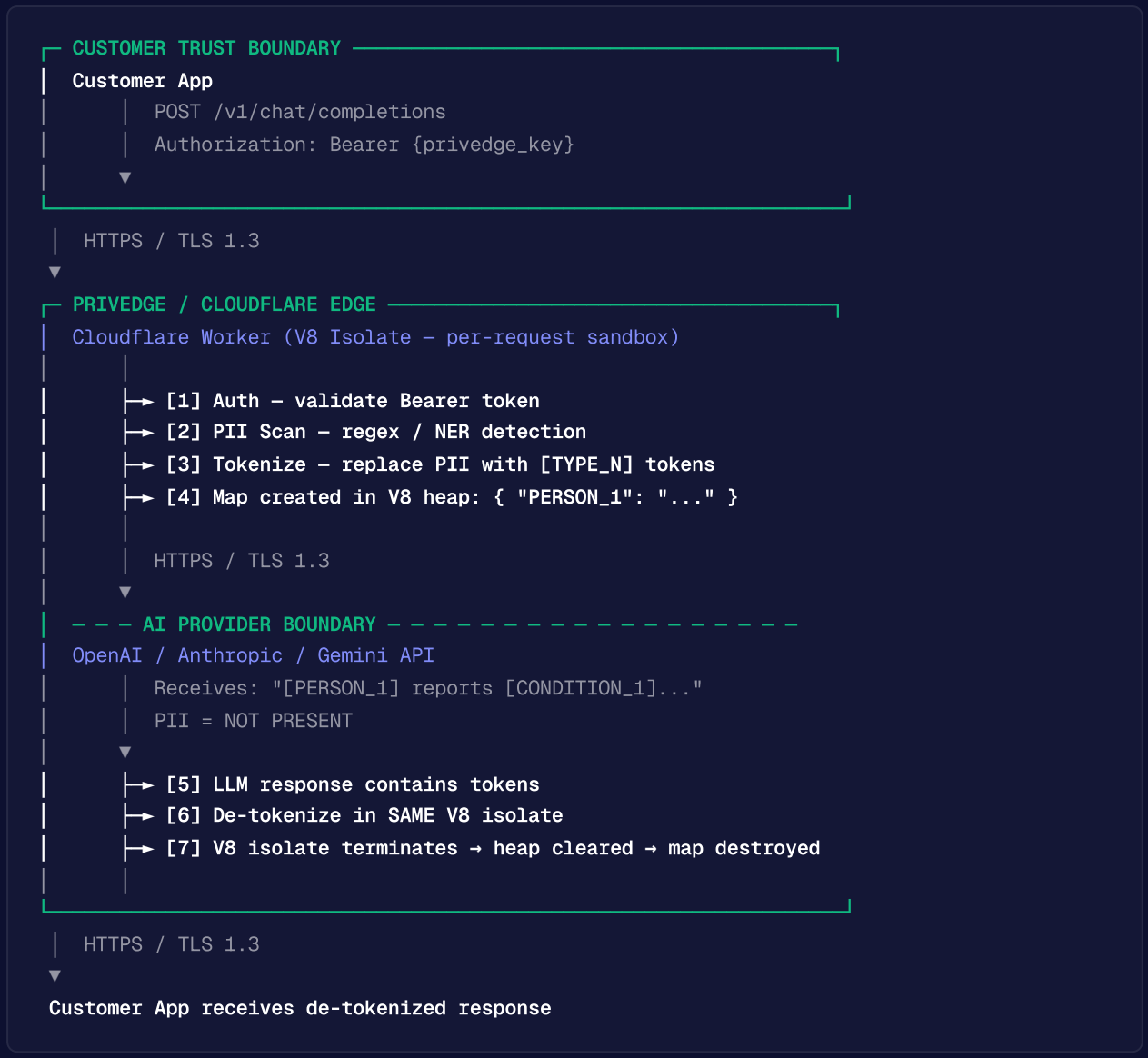
Security Engineering Reference

Request lifecycle, token map model, encryption, audit schema, and self-host architecture. For security engineering review — not for public distribution.

Cloudflare Workers | V8 Isolates | TLS 1.3 | AES-256

01 Request Lifecycle

Every AI inference request passes through the following stages. Each stage is described with its data handling properties.



02 Token Map Lifecycle

The token map is the only artifact linking synthetic tokens to real PII. Its lifecycle is constrained by the V8 isolate execution model.

```
// Pseudocode: token map lifecycle within a single V8 isolate

async function handleRequest(request: Request): Promise<Response> {
  // V8 isolate starts fresh — zero shared state with any prior request
  const tokenMap = new Map<string, string>()  // lives ONLY in this heap

  const body = await request.json()
  const { anonymized, map } = piiScanner.tokenize(body.messages, tokenMap)
  // tokenMap: { "PERSON_1": "María Ortega", "ID_1": "12345678A" }
  // anonymized: "Hello [PERSON_1], your ID is [ID_1]..."

  const llmResponse = await forwardToProvider(anonymized)
  // llmResponse: "Dear [PERSON_1], we have reviewed [ID_1]..."

  const restored = piiScanner.detokenize(llmResponse, tokenMap)
  // restored: "Dear María Ortega, we have reviewed 12345678A..."

  // After return: isolate GC'd → heap freed → tokenMap = gone
  // No KV write. No DB write. No log of PII values.
  return new Response(JSON.stringify(restored))
}
```

> **Security property:** The token map is bound to a single V8 isolate, which is bound to a single HTTP request. There is no mechanism by which the token map can be accessed after the response is returned — not by Privedge engineers, not by Cloudflare, and not by any external attacker.

---

03 Data Flow & Trust Zones

ZONE	WHAT ENTERS	WHAT EXITS	PII PRESENT?
Customer app	User input (raw)	Prompt with PII	Yes — customer's responsibility
Privedge edge (Worker)	Prompt with PII	Anonymized prompt	In-memory only, sub-millisecond
AI Provider	Anonymized prompt (tokens)	Response with tokens	No — never receives real PII
Privedge audit log	Request metadata	—	No — metadata only

04 Encryption

In transit

- All connections: TLS 1.3 minimum. TLS 1.2 disabled on Cloudflare Workers by default.
- Customer app → Privedge edge: TLS 1.3, certificate managed by Cloudflare
- Privedge edge → AI Provider: TLS 1.3 (enforced by Cloudflare egress)
- HSTS: `max-age=31536000; includeSubDomains; preload`

At rest

- Prompt content: never written to storage — no at-rest encryption needed
- Audit logs (Enterprise, Cloudflare R2): AES-256-GCM, keys managed by Cloudflare KMS
- Customer API keys: encrypted in Cloudflare Workers Secrets (AES-256)

05 V8 Isolate Sandbox

Privedge runs on Cloudflare Workers, which uses V8 — the same JavaScript engine as Chrome — in a heavily sandboxed environment.

- **Process isolation:** Each Worker invocation runs in its own V8 isolate context
- **No shared memory:** Isolates cannot access each other's heap — zero cross-request state leakage
- **No file system:** Workers have no access to a traditional file system
- **No persistent globals:** Global variables reset on cold starts
- **CPU time limits:** Maximum 50 ms CPU time per request
- **Memory limits:** 128 MB per isolate; no swap, no persistence after GC
- **Network egress controlled:** Only explicitly configured outbound destinations reachable

> **Important:** Cloudflare Workers uses V8 isolates, NOT containers or VMs. Startup ~0 ms. Security model analogous to browser tab isolation — each request is its own sandbox with no shared state.

---

06 Audit Log Schema

For Pro and Enterprise tiers, Privedge writes a metadata record per request. This record contains zero prompt content and zero PII values.

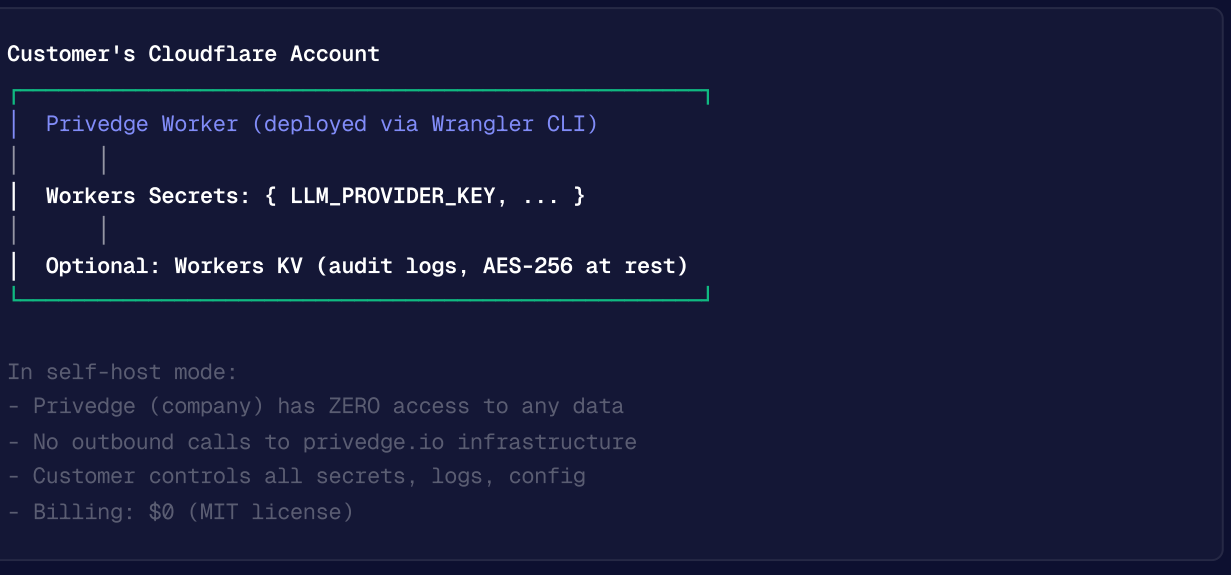
```
// AuditLog schema — stored in R2 (Enterprise) or 30-day rolling (Pro)
interface AuditLog {
  request_id:   string    // UUID — no PII
  timestamp:    string    // ISO 8601
  customer_id:  string    // hashed
  routed_to:    "openai" | "anthropic" | "gemini" | ...
  pii_types:    string[]  // ["PERSON","EMAIL"] — categories, NOT values
  pii_count:    number
  latency_ms:   number
  compliance:   string    // "gdpr" | "hipaa" | "ccpa" | "lgpd"
  model:        string
  region:       string    // Cloudflare colo: "MAD" | "FRA" | "IAD"

  // NEVER logged: prompt_content, response_content, pii_values, user_id
}
```

---

07 Self-host Architecture

Privedge is MIT licensed. Organizations with strict data sovereignty requirements deploy the Worker in their own Cloudflare account.




```
# 1. Clone the MIT-licensed repo
git clone https://github.com/privedge/privedge

# 2. Set your LLM provider secret
wrangler secret put LLM_PROVIDER_KEY

# 3. Deploy to your Cloudflare account
wrangler deploy

# 4. Point your app at your Worker URL
#    PRIVEDGE_BASE_URL=https://privedge.your-zone.workers.dev
```

> **Data sovereignty:** In self-host mode all processing occurs in the customer's own Cloudflare account. Privedge (the company) is not a sub-processor in this deployment model.