KiloClaw Security

Security Architecture, Tenant Isolation, and Data Protection for Managed AI Agent Compute

February 2026 v1.0; independent security assessment by Andrew Storms

KiloClaw is a managed compute platform that runs per user AI agent instances inside dedicated virtual machines. Each customer gets an isolated environment where their AI agent can execute code, access the filesystem, browse the web, and connect to chat channels such as Telegram, Discord, and Slack.

This category of product, managed compute for AI agents that execute arbitrary code on behalf of customers, carries inherently high security stakes. A failure in tenant isolation could expose one customer's secrets, conversations, and connected accounts to another. A failure in secret management could compromise API keys that customers entrust to the platform.

This white paper describes the security architecture of KiloClaw, the controls that protect customer data, and the results of an independent security assessment conducted in February 2026. It is intended for security teams, compliance officers, and technical decision-makers evaluating KiloClaw for their organization.

Assessment Summary

An independent 10-day security assessment validated KiloClaw's architecture through threat modeling (PASTA framework, 30 threats across 13 assets), code review, 60+ adversarial tests, and live infrastructure testing. The overall finding: KiloClaw's security architecture is sound, with tenant isolation enforced at multiple independent layers.

How KiloClaw Works

Understanding the security model requires understanding the architecture. When a customer provisions a KiloClaw instance, the platform creates a dedicated compute environment through several layers of infrastructure.

Request Flow

A customer’s browser connects to a Cloudflare Worker, which authenticates the request via a signed JWT cookie. The Worker looks up the customer’s dedicated virtual machine through a Durable Object (a per-user state store), then proxies the request to the customer’s machine on Fly.io. Inside the machine, a controller process routes traffic to the OpenClaw AI agent runtime.

Every hop in this chain enforces authentication. No user-controlled value influences which machine receives traffic, machine routing is always derived from the authenticated user identity stored server-side.

Compute Model

Each customer runs inside a dedicated Firecracker micro VM on Fly.io. Firecracker is the same virtual machine monitor used by AWS Lambda and AWS Fargate, providing hardware-level

isolation between tenants. Each customer also gets a dedicated Fly application with its own isolated network, and a dedicated NVMe backed volume for persistent storage.

This is not container-based isolation. Each customer’s workload runs in its own virtual machine with its own kernel, separated from every other customer at the hypervisor level.

Tenant Isolation

Tenant isolation is the most critical security property of any multitenant platform. KiloClaw enforces isolation at five independent layers, each of which would need to fail simultaneously for a cross-tenant breach to occur.

Layer 1: Identity-Based Routing

Every request is authenticated via a JWT cookie that is validated against both a signing secret and a per-user pepper stored in the database. The pepper enables server-side session revocation, if the pepper is rotated, all existing sessions for that user are immediately invalidated. The system fails closed: if the database is unreachable, authentication fails rather than falling back to a less secure mode.

The authenticated user identity is used to derive a deterministic Durable Object key, which maps to exactly one per-user state store. This state store holds the Fly machine and volume identifiers for that user's instance. No user-supplied value can influence which machine receives the proxied request.

Layer 2: Per-User Fly Applications

Each customer's virtual machine runs inside a dedicated Fly application. The application name is derived from a SHA-256 hash of the user identity, truncated to 80 bits of entropy. This provides a negligible collision probability, and an ownership verification check guards against the astronomically unlikely case of a hash collision.

Per-user applications are significant because Fly.io scopes resources—machines, volumes, networks—to the application level. A volume in one application cannot be mounted by a machine in another application. A machine in one application cannot be addressed by internal DNS from another application.

Layer 3: Network Isolation

Each per-user Fly application is configured with an isolated WireGuard network mesh. This means customers are separated at the network layer, not just at the application layer. Live cross-tenant testing confirmed that

internal DNS resolution fails across applications, direct IPv6 connectivity is blocked between applications, and only self-referencing network operations succeed.

This was validated through eight live tests across two separate per-user Fly applications during the security assessment.

Layer 4: Firecracker VM Boundary

Each customer's workload executes inside a Firecracker micro-VM. Firecracker provides hardware-virtualization-based isolation using KVM, with a minimal device model and a restrictive seccomp filter applied at the host level. This is the same isolation technology that underpins AWS Lambda,serving billions of invocations.

The VM boundary means that even if a customer's AI agent were compromised through prompt injection or malicious tool use, the blast radius is contained to that customer's own virtual machine. There is no shared kernel, no shared filesystem, and no shared process namespace between tenants.

Layer 5: Volume Isolation

Each customer's persistent storage is a dedicated Fly Volume (NVMe-backed block storage) that can only be mounted within the customer's own Fly application. Volumes are encrypted at rest using LUKS (AES-XTS). The volume binding chain was audited and confirmed sound—there is no path by which one customer's volume could be mounted by another customer's machine.

Adversarial Testing Results

The security assessment subjected the tenant isolation chain to 35 adversarial input tests including Unicode characters (CJK, emoji, Arabic, mixed scripts), zero-width characters, null bytes, control characters, injection payloads (path traversal, SQL, XSS, template injection), and boundary-length inputs. All were handled correctly with no cross-tenant impact.

Data Protection

Encryption in Transit

All external communication uses TLS. Browser-to-platform traffic terminates at Cloudflare's edge. Platform-to-Fly.io traffic uses TLS to the Fly proxy. AI provider API calls (Anthropic, OpenAI, etc.) use HTTPS with the customer's own API key.

Encryption at Rest

Fly Volumes are encrypted at rest using LUKS with AES-XTS. This encryption is managed by Fly.io at the infrastructure level and is transparent to the application. Customer secrets stored in the platform database are encrypted using RSA-OAEP with AES-256-GCM, a modern authenticated encryption scheme.

Data Classification

Data Type	Protection	Where Stored
API keys (customer-provided)	RSA-OAEP + AES-256-GCM encryption in database; decrypted only at machine runtime	Encrypted in Postgres; decrypted in VM environment
Chat channel tokens	Same encryption scheme as API keys	Encrypted in Postgres; decrypted in VM environment
Session transcripts	LUKS volume encryption at rest; per-user VM isolation	Fly Volume inside customer's VM
Agent workspace files	LUKS volume encryption at rest; per-user VM isolation	Fly Volume inside customer's VM
Authentication tokens	HMAC-derived; stored in per-user Durable Object	Cloudflare Durable Object state

Data Lifecycle and Deletion

When a customer destroys their KiloClaw instance, the platform executes a two phase deletion that is crash safe with alarm-based retry. The Fly machine is stopped and destroyed, the Fly volume is deleted (triggering LUKS key destruction), and all runtime secrets are removed. This process

was reviewed and confirmed to reliably clean up all sensitive data even in the face of partial failures.

A data residency review identified nine distinct data stores and confirmed that no secrets persist after instance destruction. Non-sensitive metadata (user identifiers, timestamps) may persist in soft-deleted database records and operational logs, consistent with standard practices for auditability.

Authentication and Access Control

User Authentication

KiloClaw authenticates users via JWT cookies with the following properties:

- Algorithm pinned to HS256, preventing algorithm confusion attacks
- Per-user pepper enables server-side session revocation without secret rotation
- Fail-closed design: if the database is unreachable, authentication rejects all requests
- Cookies are HttpOnly, Secure, and SameSite=Lax
- Constant-time comparison for all secret validation, preventing timing attacks

Internal API Security

Communication between KiloClaw's internal services is authenticated using shared secrets with constant-time comparison. Platform API routes are separated from user-facing routes and are not externally accessible. The security assessment confirmed there are no SQL injection, XSS, command injection, or path traversal vulnerabilities in any API endpoint.

Gateway Authentication

Each customer's AI agent instance is protected by a gateway token derived via HMAC from a per-instance secret. This token must be presented to establish WebSocket connections or make API calls to the agent. Even if a

customer's machine were directly addressable (bypassing the Cloudflare proxy), the gateway token provides an independent authentication barrier.

Input Validation and Injection Prevention

The platform implements comprehensive input validation at every layer where user-controlled data enters the system.

Attack Vector	Mitigation	Validation Result
SQL injection	Parameterized queries with type-enforced placeholders across all database operations	No findings
Cross-site scripting (XSS)	HTML entity escaping covering all five critical characters; JSON.stringify for JavaScript contexts	No findings
Command injection	Environment variable names validated against strict regex; values escaped using POSIX quoting; process spawning uses array arguments (no shell interpolation)	No findings (55 adversarial payloads tested)
Path traversal	No user-controlled file paths in the platform layer	No findings
Open redirect	All redirect URLs are server-derived, never from user input	No findings
Schema validation	All API request bodies validated through Zod schemas before processing	No findings

The command injection surface received particular attention because KiloClaw constructs shell commands to configure customer environments. Seventeen adversarial tests with 55 injection payloads for values and 30 payloads for variable names confirmed that the escaping chain is sound.

Virtual Machine Security

Firecracker Isolation

Each customer's workload runs in a Firecracker micro-VM. Firecracker provides strong isolation guarantees that have been validated at massive scale by AWS. Key properties relevant to KiloClaw:

- Hardware-virtualization-based isolation using KVM (not containers)
- Minimal virtual device model reduces the hypervisor attack surface

- Host-level seccomp filter restricts system calls available to the VMM process
- No Fly.io API token is present inside customer VMs, limiting blast radius if a VM is compromised
- The Fly.io metadata API endpoint is blocked from within customer VMs

Resource Controls

Customer VMs are provisioned with defined CPU and memory limits. The platform enforces bounds on machine sizing (CPU count and memory allocation) and allocates exactly one machine and one volume per customer. Volume sizes are fixed. These controls prevent resource abuse and ensure predictable isolation between tenants.

Agent Security Controls

The AI agent running inside each VM has configurable security controls. By default, the exec tool (which allows the agent to run shell commands) requires explicit user approval before execution. These settings are locked in the platform's start script and cannot be overridden by the agent itself or by prompt injection through chat channels.

Build and Supply Chain Security

KiloClaw's machine images are built through a controlled CI/CD pipeline with the following safeguards:

- Docker images are tagged with git commit SHAs, providing full traceability from deployed image to source code
- The OpenClaw AI runtime version is pinned in the Dockerfile (not set to "latest"), preventing unexpected version changes
- Image references in the deployment pipeline are SHA-pinned, ensuring the exact reviewed image is deployed
- All customer machines pull from a single shared image registry; per-user registries are inert and cannot influence image selection

The security assessment identified areas for further investment in supply chain hardening, including base image pinning to SHA-256 digests, image

signing with Sigstore/cosign, SBOM generation, and automated vulnerability scanning in CI. These are planned for the next phase of security work.

Network Security

Perimeter

All customer traffic passes through Cloudflare, which provides DDoS mitigation, WAF rules, and rate limiting at the edge. The Cloudflare Worker performs JWT authentication before proxying any request to the backend infrastructure.

Internal Network

Per user Fly applications create isolated WireGuard network meshes. This isolation was validated through live testing:

- Internal DNS (.internal) resolution fails across applications—one customer cannot discover another's machines
- Direct IPv6 connectivity is blocked between per-user networks
- Only self-referencing operations succeed within a customer's own network

Eight live cross-tenant network tests were conducted across two separate per-user Fly applications. All confirmed that network-level isolation is enforced.

Egress

Customer AI agents require outbound internet access to perform their work, calling APIs, fetching web pages, and using tools. Static egress filtering is not practical for this use case because agent behavior is inherently unpredictable. Outbound access is unrestricted, which is an accepted risk documented in the threat model. Egress monitoring and interactive approval controls are on the product roadmap.

Security Assessment Methodology

The security assessment was conducted over 10 days in February 2026 by an independent assessor using the PASTA (Process for Attack Simulation and Threat Analysis) framework. The assessment covered:

Phase	Scope	Result
Threat modeling	PASTA framework: 30 threats cataloged across 13 assets (6 customer, 7 platform)	Complete
Tenant isolation	Routing chain code review, 35 adversarial input tests, 8 live network isolation tests	Sound. no cross-tenant path found
Auth and access control	JWT validation, cookie security, internal API auth, proxy chain auth, controller auth	Sound. fail-closed, timing-safe
Input validation	SQL injection, XSS, command injection (72 adversarial payloads), path traversal, open redirect	Comprehensive. no findings
VM and machine security	Dockerfile review, Firecracker analysis, runtime assessment, resource limits, destroy lifecycle	Solid—image hardening in progress
Supply chain	CI/CD pipeline, image registry, version pinning, dependency analysis	Partially complete—signing and scanning planned

The assessment produced 12 review documents, authored 17 merged pull requests (10 security fixes, 7 hardening improvements), and reviewed 18 additional pull requests for security implications.

Frequently Asked Questions

Can Kilo see my API keys?

API keys that you provide are encrypted in the platform database using RSA-OAEP with AES-256-GCM. They are decrypted only when your virtual machine starts, at which point they exist in your VM’s environment. Kilo’s platform infrastructure handles encrypted keys; the decrypted keys are only present inside your isolated VM. Future enhancements include short-lived token exchange and in-memory secret stores to further reduce the exposure window.

Can other tenants access my data?

No. Your data is protected by five independent isolation layers: identity-based routing, a dedicated Fly application, an isolated WireGuard network, a dedicated Firecracker VM, and a dedicated encrypted volume. The security assessment validated this through code review, 35 adversarial input tests, and 8 live cross-tenant network tests. No cross-tenant access path was found.

Can the AI agent escape its VM?

Firecracker provides hardware-virtualization-based isolation, the same technology used by AWS Lambda and Fargate at massive scale. A VM escape would require a vulnerability in the Firecracker hypervisor itself. Additionally, even if a VM were compromised, the per-user network isolation prevents lateral movement to other customers' machines, and no Fly.io API tokens are present inside customer VMs.

What happens to my data when I delete my instance?

Instance deletion triggers a two-phase, crash-safe cleanup. Your Fly machine is destroyed, your Fly volume is deleted (which triggers LUKS encryption key destruction), and all runtime secrets are removed. A data residency review confirmed that no secrets persist after deletion. Non-sensitive metadata may remain in soft-deleted database records for operational auditability.

Can someone hijack my agent through prompt injection?

The AI agent's exec tool requires explicit user approval before executing any shell command. This setting is enforced by the platform and cannot be overridden by the agent, by prompt injection, or through chat channel messages. Even if a prompt injection succeeded in changing the agent's behavior, the blast radius is contained to your own VM—it cannot access other tenants' resources or escape the Firecracker boundary.

What if Kilo's own infrastructure is compromised?

The threat model includes compromise scenarios for every platform secret. The most sensitive credential (the Fly.io API token) is not present inside any customer VM. Platform secrets are stored in Cloudflare Worker secrets and are accessible only to the orchestration layer. The security

assessment recommended scoped credentials and per-service authentication to further reduce blast radius, and these are on the architectural roadmap.

Conclusion

KiloClaw's security architecture reflects the understanding that managed compute for AI agents is an inherently high-risk product category that demands defense in depth. The platform enforces tenant isolation at five independent layers, implements fail-closed authentication with server-side revocation, validates all inputs against injection attacks, and leverages battle-tested Firecracker virtualization for workload isolation.

An independent security assessment validated these controls through threat modeling, code review, adversarial testing, and live infrastructure testing. The assessment found the architecture to be fundamentally sound, with clearly documented areas for continued investment in image hardening, supply chain security, and operational maturity.

For questions about KiloClaw's security architecture or to request additional documentation, please contact the Kilo security team (security@kilocode.ai).