The Broker (Opaque Handle Pattern)

Achieving Deterministic Security in Probabilistic Systems

Technical Series Vol. 1

www.HeartBeatAgents.com

I. Abstract

As autonomous AI agents move from "chatbots with tools" to "representatives with authority," they require access to high-value credentials (OAuth tokens, API keys). Conventional architectures expose these credentials within the LLM's context window, creating a catastrophic vulnerability to Prompt Injection. Heart Beat Agents introduces the Broker (Opaque Handle pattern): a "Security by Architecture" approach that ensures the LLM never possesses, sees, or processes raw credentials. By replacing identity with ephemeral, non-informational pointers, the Heart Beat Agents architecture achieves a fail-closed security posture that does not rely on model alignment or prompt-based guardrails.

II. The Credential Crisis: Why Policy is Not Security

The industry standard for agentic security currently relies on Security by Policy, instructing the model not to reveal its keys. However, in a non-deterministic system, any data present in the context window is extractable.

- **The Injection Vector**: Malicious instructions hidden in external data (emails, web pages) can override system prompts to exfiltrate tokens.
- **The Encryption Fallacy**: Encrypting tokens at rest is a solved problem, but "Encrypted Pass-through" fails because the decryption key must eventually exist in the same execution environment as the agent, leaving it vulnerable to compromised dependencies or sophisticated extraction.

III. The Architecture of the Broker

The Heart Beat Agents Credential Broker acts as an air-gapped intermediary between the agent's reasoning engine and the external world.

The Opaque Handle Mechanism: Instead of a token, the agent is issued an Opaque Handle, a pointer with zero informational entropy.

- **Structure**: Composed of a platform prefix (hb_cred_) and 24 bytes of cryptographically random data generated via secrets.token_urlsafe().
- **Information Content**: Zero. There is no mathematical or cryptographic relationship between the handle and the underlying token; it cannot be reversed, hashed, or decrypted.
- **Volatility**: The handle-to-token mapping exists exclusively in the local memory of a single Python worker process. It is never persisted to disk, Redis, or a database.

IV. Technical Execution: The Three-Phase Lifecycle

The security of the Broker is enforced through a strict, ephemeral lifecycle:

Phase	Technical Action	Security Outcome
1. Registration	The tool requests a handle. The Broker decrypts the real token, generates a random hb_cred_ string, and stores the mapping in a local dictionary.	The token is held in memory for seconds, never reaching the LLM context.
2. Execution	The agent passes the handle to the Broker. The Broker performs Auth Injection, inserting the token into the HTTP header and validating the destination via Egress Policy.	The Agent controls the intent (URL/Method) but the Platform controls the authority (Token/Header).
3. Revocation	Upon completion, failure, or a 300s timeout, revoke_all() is triggered.	The mapping is purged from memory; handles become permanent "dead links".

V. Auth Injection & Collision Detection

To prevent the agent from manipulating the authentication protocol, the Heart Beat Agents architecture utilizes a platform-controlled integration definition.

- **Static Definitions**: The platform, not the agent, determines if an integration uses Bearer, X-API-Key, or Query params.
- **Collision Detection**: If a compromised agent attempts to provide its own Authorization header to hijack a request, the system triggers an Auth Collision Error and immediately kills the execution.

VI. Boundary 2: Egress Policy & Method-Aware Validation

The core vulnerability in most autonomous systems is the "Open Gateway"—if the agent has a tool to "make a web request," it can theoretically exfiltrate data to any listener on the internet. The Heart Beat Agents platform replaces this with a Method-Aware Egress Policy.

- **Host-Integration Coupling**: The system maintains a strict mapping between an integration (e.g., Salesforce) and its authoritative API domains (e.g., *.salesforce.com).
- **State-Changing Restrictions**: While GET requests are generally permitted for broad information gathering, "State-Changing" methods (POST, PUT, PATCH, DELETE) are strictly restricted.
- **The Guardrail**: An agent can only send a POST payload to a host that is explicitly linked to its bound Integration Account. If an agent tries to POST sensitive GitHub data to attacker-controlled-server.com, the Egress Policy blocks the request at the Broker level before the socket even opens.

VII. Boundary 3: DNS Rebinding Defense

For an audience with high technical literacy, the Time-of-Check-to-Time-of-Use (TOCTOU) vulnerability in network requests is a primary concern. Traditional validation checks a URL, resolves the IP, approves it, and then the HTTP library resolves it again to make the actual connection. An attacker can use a DNS server with a very low TTL to swap a "safe" IP for a "private" IP (like 127.0.0.1 or 169.254.169.254) between those two moments.

The Heart Beat Agents architecture implements a “Validating Network Backend” to neutralize this:

- **Atomic Resolution**: The backend performs DNS resolution and validation at the exact moment of the TCP connection.
- **IP Pinning**: Once an IP is validated against the Egress Policy and checked for private-range overlap, the system connects directly to that IP.
- **Outcome**: This eliminates the gap that DNS rebinding attacks exploit, ensuring the agent cannot be tricked into "Server-Side Request Forgery" (SSRF) against internal infrastructure.

VIII. Boundary 4: The Egress Proxy (Container Enforcement)

As a final fail-safe for the network layer, the platform doesn't just rely on Python-level logic. We implement enforcement at the Container Network Interface (CNI) level.

Feature	Mechanism	Security Impact
Squid Proxy Routing	All outbound traffic from code execution containers is forced through a dedicated Squid proxy.	Prevents "rogue" code (e.g., malicious pip packages) from bypassing the Python Broker.
Categorical Blocking	The proxy is configured to block all POST/PUT/PATCH/DELETE requests by default.	Only explicitly white-listed integration traffic is permitted through the "hole" in the proxy.
Private Range Nulling	All RFC 1918 private IP ranges are hard-blocked at the proxy level.	Provides a second layer of defense against SSRF, even if the primary backend validation were to fail.

IX. The Interaction: Intent vs. Authority

In this architecture, a "Request" is not a single action but a multi-stage validation handshake:

1. **Agent Intent**: "I want to update a Lead in Salesforce."
2. **Broker Check**: Does this agent have an active “Agent Integration” binding for Salesforce? (Boundary 1)
3. **Policy Check**: Is the target URL a valid Salesforce API endpoint for a PATCH method? (Boundary 2)
4. **Network Check**: Does the resolved IP belong to a public, non-malicious range? (Boundary 3).
5. **Proxy Check**: Is the outbound traffic authorized by the container's network rules? (Boundary 4)

The result: Even a "fully compromised" LLM logic flow is trapped within a high-fidelity sandbox.

X. Boundary 5: The Multi-Point Scrubber

Boundary 5 assumes that despite the Broker and Egress layers, a credential or sensitive handle has somehow entered a data stream. This is the "Fail-Safe" for logging and observability.

- Compiled Regex Engine: Heart Beat Agents utilizes 11 high-performance compiled regex patterns specifically tuned for high-entropy strings (JWTs, ghp_, ya29., xoxb-, etc.) and the internal hb_cred_ prefix.
- Five-Point Interception: Unlike standard middleware that only scrubs logs, the Heart Beat Agents Scrubber is integrated into five distinct data pathways:

1. **Tool Results**: Before data returns to the LLM context.
2. **Error Messages**: Preventing stack traces from leaking secrets.
3. **Event Streams**: Real-time scrubbing of WebSocket traffic.
4. **Audit Logs**: Ensuring long-term storage is clean.
5. **LLM Context Ingestion**: A final gate before the model "reads" any input.

XI. Boundary 6 & 7: Hardware-Level Multi-Tenancy

The final boundaries shift from software logic to the Kernel and Virtualization layers, addressing the risk of cross-organization data leakage or container escapes.

Boundary 6: Validated WebSocket Auth

- All tool results streamed to the frontend undergo a secondary authentication handshake. Even if a process is compromised, the data cannot be exfiltrated via the user's active UI session without a valid, per-session verified token.

XI. Boundary 6 & 7: Hardware-Level Multi-Tenancy

Boundary 7: Hardened Docker Isolation

The execution environment is designed as a "Cellular" architecture where each organization is physically isolated at the filesystem level.

Vector	Defensive Implementation	Result
User Privileges	Runs as non-root (UID 1000).	Prevents most common container escape exploits.
System Calls	cap-drop ALL + No-new-privileges.	Strips the container's ability to escalate control over the host kernel.
Filesystem	Read-only root filesystem.	Prevents the installation of persistent malware or rootkits within the run.
Storage	Dedicated organization-specific mounts.	Cross-org file access is physically impossible; mount points do not exist for other tenants.

XII. Conclusion: The Architecture of Trust

The innovation of Heart Beat Agents lies in the transition from Probabilistic Safety (hoping the model behaves) to Architectural Determinism (ensuring the model cannot misbehave). By treating the LLM as a "Confused Deputy" and surrounding it with seven layers of deterministic boundaries, the Heart Beat Agents architecture solves the fundamental paradox of AI agency: providing total power with zero risk.

This document was authored by Jyhad Aamri, the principal architect behind the Heart Beat Agents security framework.

Jyhad Aamri
Chief Executive Officer @ Mosaic Singularity
hello@JyhadAamri.com
www.JyhadAamri.com