checkout.md — The Credential Wallet for the Open Agent Ecosystem

Strategy, Architecture & Open-Source Proposal

---

1. Executive Summary

AI agents are moving from chat assistants to autonomous actors — booking flights, fighting insurance companies, buying goods, and managing calendars on behalf of their users. The infrastructure for humans to safely delegate real-world authority to these agents does not yet exist in the open-source ecosystem.

checkout.md is an open-source, MCP-native credential wallet that gives AI agents a secure, scoped, and auditable way to carry and present credentials on behalf of their human owners. Rather than competing with enterprise protocols (Stripe's ACP, Google's UCP), checkout.md targets the exploding grassroots agent ecosystem — OpenClaw and its successors — where the credential security problem is urgent, unserved, and dangerous.

---

2. The Problem

2.1 Agents Are Acting, Not Just Talking

The agentic AI shift is no longer theoretical. OpenClaw — an open-source personal AI agent — went from zero to 116,000+ GitHub stars in weeks. Users run it 24/7 on dedicated hardware, controlling it via WhatsApp and Telegram. These agents send emails, manage calendars, browse the web, make purchases, and execute shell commands autonomously.

This pattern is replicating across Claude Code, Cursor, Manus, and dozens of emerging agent frameworks. The common thread: agents need access to real-world credentials (API keys, payment methods, identity proofs, service logins) to do real-world work.

2.2 The Security Situation Is Critical

Today's credential handling in the indie agent ecosystem is alarmingly primitive:

- **Raw secrets in config files.** OpenClaw users paste API keys, Discord tokens, and service credentials directly into YAML. Every skill and plugin gets access to everything.
- **No permission scoping.** When an agent gets email access, every skill — including third-party community skills — inherits that access with no restrictions.
- **Active exploitation.** Cisco's AI security team tested a popular OpenClaw skill and confirmed it was performing data exfiltration and prompt injection without user awareness. The skill had been inflated to #1 in the skill registry.
- **Systemic risk.** CrowdStrike published a detailed analysis showing misconfigured OpenClaw instances can become AI backdoors. The project's own maintainers have warned it is "far too dangerous" for users who don't understand the command line.

2.3 Enterprise Solutions Don't Reach the Grassroots

The enterprise world is moving fast. Stripe and OpenAI launched the Agentic Commerce Protocol (ACP) with scoped payment tokens. Google launched the Universal Commerce Protocol (UCP) with OAuth 2.0 and verifiable credentials, backed by Shopify, Walmart, Target, Visa, and Mastercard. Crossmint is building agent wallets with Apple Pay and ACH integration.

But none of these serve the indie developer running OpenClaw on a Raspberry Pi, or the power user whose agent autonomously built a website from their phone. These protocols are designed for merchant-platform integration, not personal agent security.

The grassroots agent ecosystem has zero credential infrastructure. That is the gap.

---

3. The Opportunity

3.1 Why Now

Three forces converge to make this moment uniquely viable:

1. **MCP is the universal agent bus.** Model Context Protocol has emerged as the standard interface between LLMs and external tools. OpenClaw runs on MCP. Claude Code runs on MCP. Any wallet that speaks MCP is instantly compatible with the entire ecosystem.
2. **The pain is visceral and documented.** This isn't a theoretical future risk — Cisco and CrowdStrike are publishing incident reports *right now*. Users are watching their agents leak credentials and execute malicious code. The community is actively looking for solutions.
3. **The open-source community is the distribution channel.** OpenClaw's growth was

entirely community-driven: no sales team, no enterprise contracts — just a good README, Docker support, and word of mouth. A credential wallet that follows the same pattern gets the same distribution for free.

3.2 Competitive Positioning

Layer	**Players**	**checkout.md Position**
Enterprise payment protocols	Stripe ACP, Google UCP	Not competing
Platform-specific agent wallets	Crossmint, Google Wallet	Not competing
Verifiable identity for agents	Wallet4Agent	Adjacent / complementary
Credential governance for open-source agents	**Nobody**	**This is the gap**

checkout.md is not a payment processor, not an identity provider, and not a protocol standard. It is the **governance and delegation layer** — the place where a human defines what their agent can and cannot do, and the agent proves those boundaries to any service it interacts with.

---

4. Product Concept

4.1 Core Value Proposition

“The allowance system for your AI agent.”

Just as parents give children spending money with rules, checkout.md lets humans give their agents credentials with constraints:

- Here’s access to my email — **read-only, only the inbox, not sent mail**
- Here’s a payment method — **$200/month cap, only these merchant categories, require my approval above $75**
- Here’s my calendar — **can create events, cannot delete or modify existing ones**
- Here’s my GitHub token — **only these repos, only pull requests, no force pushes**

4.2 Key Capabilities

Scoped Credential Vaulting Store credentials (API keys, OAuth tokens, payment tokens, service passwords) encrypted at rest, never exposed to the agent or its skills directly. Issue short-lived, scoped tokens on demand.

Per-Skill Permission Boundaries Define which skills/plugins can access which credentials, with what scope, and under what conditions. The "What Would Elon Do?" skill gets zero access. Your trusted email skill gets read-only inbox access.

Spending & Action Policies Programmable rules: daily/monthly spend limits, merchant category whitelists/blacklists, time-of-day restrictions, approval thresholds requiring human confirmation via the user's messaging platform.

Audit Trail Every credential request, every approval, every denial — logged and queryable. When your agent books a flight, you can trace exactly which skill requested the payment token, what scope it received, and what it did with it.

Cross-Agent Portability Credential once, authorize everywhere. One checkout.md wallet works across OpenClaw, Claude Code, Cursor, and any MCP-compatible agent — no more pasting API keys into five different config files.

Human-in-the-Loop Escalation Configurable approval flows. Low-risk actions (read email subject lines) proceed automatically. Medium-risk actions (send an email) get a notification. High-risk actions (make a purchase over $100) require explicit approval via WhatsApp/Telegram/Slack before the token is issued.

---

5. Architecture & Open-Source Approach

5.1 Design Principles

1. **Local-first.** The wallet runs on the user's own hardware, next to their agent. No cloud dependency for core functionality. Credentials never leave the user's machine unencrypted.

2. **MCP-native.** Implemented as an MCP server that any MCP-compatible agent can attach to immediately. No custom SDKs, no proprietary protocols.

3. **Zero-trust toward skills.** Skills never receive raw credentials. They receive scoped, short-lived tokens through the wallet's MCP tools. The wallet mediates every interaction.

4. **Open-source core, open standard.** The wallet, the policy language, and the MCP tool definitions are fully open-source under Apache 2.0.

5.2 System Architecture

```
┌─────────────────────────────────────────────────┐
│                   User's Device                   │
│                                                   │
│  ┌──────────────┐        ┌────────────────────┐   │
│  │ OpenClaw /    │  MCP   │  checkout.md        │   │
│  │ Claude Code   │◄──────►│  (MCP Server)       │   │
│  │ / Any Agent   │        │                     │   │
│  └──────┬───────┘        │  ┌──────────────┐  │   │
│         │                │  │ Policy Engine │  │   │
│         │                │  └──────────────┘  │   │
│  ┌──────┴───────┐        │  ┌──────────────┐  │   │
│  │ Skills /      │        │  │ Credential    │  │   │
│  │ Plugins       │        │  │ Vault (enc.)  │  │   │
│  └──────────────┘        │  └──────────────┘  │   │
│                          │  ┌──────────────┐  │   │
│                          │  │ Audit Log     │  │   │
│                          │  └──────────────┘  │   │
│                          │  ┌──────────────┐  │   │
│                          │  │ Approval Bus  │  │   │
│                          │  │ (WA/TG/Slack) │  │   │
│                          │  └──────────────┘  │   │
│                          └────────────────────┘   │
└─────────────────────────────────────────────────┘
```

5.3 Core Components

5.3.1 Credential Vault

- **Storage:** SQLite + SQLCipher (encrypted at rest) or SOPS-encrypted files for simpler setups
- **Credential types supported:** API keys, OAuth 2.0 tokens (with refresh), payment tokens (Stripe/ACP-compatible), service passwords, arbitrary secrets
- **Key management:** User-provided master passphrase, derived via Argon2id. Optional hardware key (YubiKey/TPM) for high-security setups

5.3.2 Policy Engine

A declarative YAML/TOML policy language that defines what each agent, skill, or context can access:

```
# checkout.policies.yaml
policies:
  - name: "email-readonly"
    credentials: ["gmail-oauth"]
    scope:
      actions: ["read"]
      filters: ["inbox-only"]
    grant_to:
      skills: ["email-summary", "daily-briefing"]

  - name: "shopping-allowance"
    credentials: ["stripe-payment-token"]
    scope:
      max_per_transaction: 75.00
      max_per_month: 200.00
      merchant_categories: ["retail", "groceries"]
      require_approval_above: 50.00
    grant_to:
      skills: ["shopping-agent"]
      agents: ["openclaw-main"]

  - name: "github-limited"
    credentials: ["github-pat"]
    scope:
      repos: ["myorg/frontend", "myorg/docs"]
      actions: ["pull_request.create", "issue.comment"]
      deny: ["repo.delete", "push.force"]
    grant_to:
      skills: ["code-review"]

  - name: "calendar-managed"
    credentials: ["google-calendar-oauth"]
    scope:
      actions: ["event.create", "event.read"]
      deny: ["event.delete", "event.update"]
      calendars: ["personal"]
    grant_to:
      agents: ["*"]  # all agents can read/create
```

5.3.3 MCP Tool Interface

The wallet exposes a small, focused set of MCP tools:

```
checkout_request_credential
  - policy_name: string
```

```
    - purpose: string (agent describes why it needs access)
    - context: object (skill_id, agent_id, action_details)
    → Returns: scoped_token (short-lived) | approval_pending | denied

  checkout_list_available_policies
    - agent_id: string
    → Returns: list of policy names this agent can request

  checkout_check_budget
    - policy_name: string
    → Returns: remaining budget, transaction count, next reset

  checkout_report_usage
    - token_id: string
    - outcome: success | failure | cancelled
    - details: object
    → Returns: ack (logged to audit trail)
```

5.3.4 Approval Bus

When a policy requires human approval, the wallet sends a structured approval request to the user's preferred messaging channel:

```
🔐 checkout.md — Approval Required

Agent: openclaw-main
Skill: shopping-agent
Action: Purchase — "Running shoes, Nike Air Zoom"
Amount: $89.99
Merchant: nike.com
Policy: shopping-allowance (requires approval above $50)

Reply: ✅ APPROVE or ❌ DENY
```

Integrations: WhatsApp (via OpenClaw's existing channel), Telegram, Slack, Signal, email fallback. The approval bus is itself an MCP tool, meaning agents can be configured to route approvals through whatever channel the user prefers.

5.3.5 Audit Log

Append-only local log (SQLite or structured JSONL):

```
{
  "timestamp": "2026-02-20T14:32:01Z",
  "event": "credential_issued",
  "policy": "shopping-allowance",
  "agent": "openclaw-main",
  "skill": "shopping-agent",
  "purpose": "Purchase running shoes from nike.com",
  "token_id": "tok_abc123",
  "scope": {"max_amount": 75.00, "merchant": "nike.com"},
  "approval": "auto",
  "outcome": "pending"
}
```

5.4 Technology Stack

Component	Technology	Rationale
Runtime	Node.js (TypeScript)	MCP SDK is TypeScript-first; matches OpenClaw ecosystem
Credential storage	SQLCipher / SOPS	Encrypted at rest, no cloud dependency
Policy engine	Custom YAML parser + CEL (Common Expression Language)	Declarative, auditable, Google-backed expression language
MCP server	`@modelcontextprotocol/sdk`	Official MCP SDK, first-class support
Approval notifications	Platform webhooks (WA/TG/Slack APIs)	Reuses channels agents already use
Audit log	SQLite (append-only)	Simple, portable, queryable
Packaging	Docker, npm, Nix	Matches OpenClaw's distribution patterns

5.5 Deployment Model

```
# Option 1: npm (simplest)
npm install -g @checkoutmd/wallet
checkout-wallet init
checkout-wallet serve  # starts MCP server on local socket

# Option 2: Docker (isolated)
docker run -v ~/.checkout:/data checkoutmd/wallet
```

```
# Option 3: Alongside OpenClaw
# Add to openclaw gateway config:
mcp:
  servers:
    checkout:
      command: checkout-wallet serve
      args: ["--vault", "~/.checkout/vault.db"]
```

---

6. Go-to-Market: Community-First

6.1 Launch Sequence

Phase 1 — OpenClaw Skill (Weeks 1–4) Ship a working OpenClaw skill that wraps checkout.md. Users install it like any other skill. Immediate value: API keys and tokens move out of plaintext config into an encrypted vault with basic scoping. Publish to ClawHub.

Phase 2 — Standalone MCP Server (Weeks 4–8) Release the full MCP server as an independent package. Works with OpenClaw, Claude Code, Cursor, or any MCP client. Policy engine, approval flows, and audit log. Docker and npm distribution.

Phase 3 — Ecosystem Integrations (Months 2–4) Provide pre-built policy templates for common services: Gmail, GitHub, Stripe, Google Calendar, Shopify, AWS. Publish as a community-maintained "policy registry" on GitHub.

Phase 4 — Trust Signals (Months 4–6) Work with the OpenClaw foundation (post-Steinberger) and other agent framework maintainers to establish "checkout.md compatible" as a trust signal for skills. Skills that use the wallet for credential access can be flagged as safer than those that demand raw secrets.

6.2 Distribution Channels

- **GitHub:** Open-source repo, Apache 2.0
- **ClawHub:** OpenClaw skill registry
- **npm:** `@checkoutmd/wallet`
- **Docker Hub:** `checkoutmd/wallet`
- **OpenClaw Discord / community:** Direct engagement with the 100k+ user base

6.3 Business Model (Optional)

The core wallet is free and open-source forever. Potential revenue:

- **checkout.md Cloud:** Hosted vault with multi-device sync, backup, and team features for small businesses whose agents act on shared credentials
- **Enterprise policy management:** Centralized policy administration for orgs with many agents/employees
- **Compliance & audit tooling:** Export-ready audit logs for regulated industries
- **Premium integrations:** Pre-built connectors for enterprise services (Salesforce, SAP, etc.)

---

7. Risks & Mitigations

Risk	Mitigation
OpenClaw builds credential management natively	Be protocol-level (MCP), not platform-specific. Work across all agents.
OpenAI / Anthropic absorb the space with first-party solutions	First-party solutions will be platform-locked. Open-source, local-first is the counter-position.
Standards fragmentation (ACP vs UCP vs custom)	Wallet is credential-type agnostic — it stores and scopes whatever tokens the user has, regardless of which protocol issued them.
Low adoption / chicken-and-egg	Ship as an OpenClaw skill first — instant access to 100k+ users who have the problem today. No chicken-and-egg.
Security of the wallet itself becomes an attack vector	Open-source for audit, SQLCipher for encryption, minimal attack surface (local socket only, no network exposure by default), optional hardware key support. Bug bounty program.
Peter Steinberger joining OpenAI shifts OpenClaw's trajectory	checkout.md is MCP-native, not OpenClaw-native. It works with whatever comes next.

---

8. Summary

The agentic AI revolution has a security crisis hiding in plain sight. Millions of autonomous agents are about to operate on behalf of humans, and the open-source ecosystem has no standard way to manage the credentials these agents need. Enterprise players are solving this for their own platforms, but the fastest-growing, most innovative part of the agent ecosystem — the indie, open-source, run-it-yourself community — is flying blind.

checkout.md fills that gap as an open-source, MCP-native credential wallet with scoped permissions, policy-driven governance, human-in-the-loop approvals, and full audit trails. It meets users where they are (OpenClaw, Docker, npm), speaks the language they already use (MCP, YAML policies), and solves a problem they are experiencing right now (credential leaks, malicious skills, zero access control).

The edge isn't technology. The edge is positioning: be the open-source credential standard for the agent ecosystem before the platforms lock it down.

---

checkout.md — Because your AI shouldn't have your credit card without a spending limit.