Cred402 — SpendGuard

A protocol-enforced spending limit for AI agents, on NEAR

NEAR Hackathon 2026 — Agent Economy track

github.com/caelum0x/cred402

The problem

AI agents now hold real wallets and spend real money — autonomously.

- An agent can call an API, pay for compute, or pay another agent, with **no human in the loop**.
- Every safeguard shipped so far lives in **application code**: a budget variable, a try/catch, a rate limiter.
- Application code can be bypassed, bugged, or bypassed by the same model that wrote it.

If the agent is compromised, buggy, or just wrong — nothing at the protocol level stops it from spending past its limit.

Who has this problem, today

- **Agent-economy platforms** letting agents pay for APIs, compute, or data (this hackathon's own track brief).
- **API/data sellers** who want to sell to agents per-call, without issuing API keys to humans.
- **Marketplaces** that let one agent hire another and need a real trust signal before extending credit.
- **Anyone wiring an LLM to a wallet** — the exact audience NEAR's Chain Signatures and Intents are built for.

The solution

Cred402 sells a pay-per-call credit-score API for AI agents over **x402** (`402 Payment Required`). The same codebase already has working Algorand and QIE x402 routes (both currently report `configured: false` in production — pay-to address not set, so neither is actually live yet) — this submission adds **native NEAR** as a third settlement rail, and the only one of the three with a public, re-verifiable on-chain proof trail.

SpendGuard (new, built in this hackathon window) turns that score into a spending limit **nearcore itself enforces**:

```
earn (verified x402 revenue)
  -> credit policy -> global exposure cap
  -> a real NEAR Function Call Access Key, allowance = credit line
  -> agent spends through that key
  -> a dispute-rate spike freezes exposure AND revokes the key
```

Not a config flag. Not a `require!()`. A real NEAR access-key allowance.

Live proof — 25 real spends, repay, top-up (not a simulation)

```
  { attempt: 6, success: true, detail: 'settled' },
  { attempt: 7, success: true, detail: 'settled' },
  { attempt: 8, success: true, detail: 'settled' },
  { attempt: 9, success: true, detail: 'settled' },
  { attempt: 10, success: true, detail: 'settled' },
  { attempt: 11, success: true, detail: 'settled' },
  { attempt: 12, success: true, detail: 'settled' },
  { attempt: 13, success: true, detail: 'settled' },
  { attempt: 14, success: true, detail: 'settled' },
  { attempt: 15, success: true, detail: 'settled' },
  { attempt: 16, success: true, detail: 'settled' },
  { attempt: 17, success: true, detail: 'settled' },
  { attempt: 18, success: true, detail: 'settled' },
  { attempt: 19, success: true, detail: 'settled' },
  { attempt: 20, success: true, detail: 'settled' },
  { attempt: 21, success: true, detail: 'settled' },
  { attempt: 22, success: true, detail: 'settled' },
  { attempt: 23, success: true, detail: 'settled' },
  { attempt: 24, success: true, detail: 'settled' },
  { attempt: 25, success: true, detail: 'settled' }
]

Repaying 25 yoctoNEAR of real outstanding exposure for demo-buyer-1794013870339...
```

`spendguard-024e8dd6f7.testnet`, public NEAR Testnet — re-verifiable with `npm run demo:near-spend-guard`.

Live proof — repay → top-up → draw again → freeze

```
  { attempt: 25, success: true, detail: 'settled' }
]

Repaying 25 yoctoNEAR of real outstanding exposure for demo-buyer-1794013870339...
exposure after real repay: {
  agent_id: 'demo-buyer-1794013870339',
  outstanding: '25',
  reserved: '299999999999999999999975',
  max_allowed: '300000000000000000000000',
  frozen: false,
  updated_at: 1794013917736209200
}
Topping up demo-buyer-1794013870339 by the same 25 yoctoNEAR on its SAME live key...
exposure after real top-up: {
  agent_id: 'demo-buyer-1794013870339',
  outstanding: '0',
  reserved: '300000000000000000000000',
  max_allowed: '300000000000000000000000',
  frozen: false,
  updated_at: 1794013920668150800
}
real spend after top-up (must succeed, on the ORIGINAL key): [ { attempt: 1, success: true, detail: 'settled' } ]
Authorized demo-scammer-1794013924305; simulating a dispute-rate spike...
```

A real `repay_credit_line` zeroes outstanding exposure; a real `top_up_allowance` re-mints the **same live key** with restored headroom.

Live proof — nearcore rejects the frozen key itself

```
}
real spend after top-up (must succeed, on the ORIGINAL key): [ { attempt: 1, success: true, detail: 'settled' } ]
Authorized demo-scammer-1794013924305; simulating a dispute-rate spike...
frozen: true
key_active: false
post-freeze spend attempt (must fail): [
  {
    attempt: 1,
    success: false,
    detail: 'Access key ed25519:ALMnhuytsPY1eG3i6vQNk1YW6qSJKKLc8W3LbGz8Tf6R does not exist at block height 276563295'
  }
]

Live Testnet loop verified for real: {
  contract: 'spendguard-024e8dd6f7.testnet',
  rpc: 'https://rpc.testnet.fastnear.com',
  buyer_credit_line: '300000000000000000000000',
  real_spends: 25,
  revolving_credit_proved: true,
  scammer_frozen: true
}

demo:near-spend-guard complete.
```

A dispute spike freezes the key — the next spend fails with **`Access key ... does not exist`**, NEAR's own protocol error. Re-confirmed live today (2026-11-11) via direct RPC to `spendguard-024e8dd6f7.testnet`.

How it works

Layer	What it is
`near_gateway.ts`	From-scratch x402 v2 `exact` scheme for NEAR — no `@x402/near` package exists yet
`near_finality.ts`	**No third-party facilitator** — Cred402 reads the payer's tx back from NEAR RPC itself
`spend_guard/*.rs`	`policy` (credit line) → `exposure` (global cap) → `spend_key` (real access key) → `scam_detection` (circuit breaker)
Tests	18/18 Rust + 14/14 near-workspaces + 450/450 full suite — **450/450 re-run today, 11-11**

Architecture detail in `docs/near_x402.md` and `docs/near_spend_guard.md`.

Why Agent Economy, and why this scores well

> Track 07 brief: *"Give AI a wallet. Build agents that earn, spend, negotiate, hire other agents and pay for APIs, compute or data — with crypto directly or through NEAR Intents."*

Cred402 **is** that: an agent calling `GET /v1/x402/near/credit-score/:agentId`, paying per call in native NEAR, no subscription, no human-issued API key.

Judging criteria	How this answers it
Technical	Real contract, real nearcore quirk found & solved (`AddKeyAlreadyExists`), 482 tests green
Design	Credit line *is* the access key — enforcement lives in the protocol, not app code
Impact	Reusable pattern: any x402 seller can rate-limit agent spend, trustlessly
Originality	No `@x402/near` exists; no project turns a credit score into a NEAR key allowance
Sustainability	Self-facilitated; Chain Signatures / Intents extension points documented, not mocked

Traction — honest numbers

- **$0 revenue.** This is hackathon-window infrastructure, not a live SaaS yet.
- **What's real:** a public GitHub repo (pushed 2026-11-10), a live public Testnet contract with 25 real on-chain spends + a real repay + a real top-up + a real freeze, a published X demo post, 450/450 tests green today.
- The same credit-score codebase also has **Algorand** and **QIE** x402 routes deployed (`cred402-1.onrender.com`, currently `configured: false` — pay-to address not set, not yet production-live); **NEAR is the third chain, and the only one with a public, re-verified on-chain proof trail** (25 real spends + repay + top-up + freeze), not a one-off demo.

Roadmap

- **Near-term:** NEAR mainnet deployment once the contract has an external audit pass.
- **Chain Signatures:** let an agent with a key on *any* chain sign the same NEAR `Transfer` / `spend_to_seller` calls via NEAR's MPC network — architecture already compatible, not yet wired.
- **NEAR Intents:** swap any asset into the exact NEAR amount before paying — same extension point.
- **Facilitator-less x402-on-NEAR as a reference implementation** other sellers can adopt directly.

Team & links

Solo-built, shipped across three chains (Algorand, QIE, NEAR) in one hackathon window.

- Repo: **github.com/caelum0x/cred402**
- Live contract: **spendguard-024e8dd6f7.testnet** (NEAR Testnet)
- Demo video: `docs/near-hackathon/media/cred402-near-spendguard-pitch-demo.mp4`
- 17s raw terminal proof: `media/cred402-near-spend-guard-demo.mp4`
- X demo post: **x.com/caelum0x42/status/2120268547800678911**

Thank you