MindVault

Non-Custodial Vault Protocol for AI Trading Agents

Technical Whitepaper v1.0

March 2026

Grim Labs | mindvault.nexus

1. Abstract

MindVault is a non-custodial vault protocol that enables AI agents to trade on-chain without holding private keys or custodying user funds. The protocol enforces safety guardrails at the smart contract level — token whitelists, daily spending limits, role-based permissions, and auto-expiration — ensuring that even a compromised agent cannot exceed its configured constraints. Orders are executed through an RFQ-style solver competition model where external fillers compete for best execution via the open IOrderFiller interface.

MindVault is built by the core team behind SpookySwap, one of the longest-running DEXs in DeFi with $66.5B lifetime volume and $1.2B peak TVL, and is designed to be the security infrastructure layer for the emerging AI-agent trading ecosystem.

2. Problem Statement

AI agents managing on-chain capital represent one of the fastest-growing segments of DeFi. Telegram trading bots collectively process billions in volume. Agentic wallets from major platforms are launching. AI fund managers and copy-trading systems are proliferating. Yet every current solution shares a fundamental architectural flaw: the agent holds full custody of user funds.

2.1 Current Landscape

Telegram trading bots (Banana Gun, Maestro, BONKbot): Users hand over private keys or use bot-generated wallets with no on-chain constraints. One exploit drains everything.

Coinbase Agentic Wallets: Custodial solution with policy-enforced limits. Safety depends on Coinbase's platform rules, not immutable smart contracts. Policies can change, servers can be compromised.

HeyAnon and similar AI agents: Unlimited wallet access with no spending limits, token whitelists, or expiration. The agent can do anything the wallet owner can do.

Traditional DeFi vaults (Yearn, Beefy): Automated strategies but no AI agent flexibility. Users pick from fixed strategies; they cannot authorize their own agents or customize guardrails.

2.2 The Cost of No Guardrails

On March 12, 2026, a trader lost approximately $50 million in a single transaction on the Aave protocol. The trade swapped $50.4M in aEthUSDT for aEthAAVE through a SushiSwap pool holding only $73,000 in liquidity, resulting in 99% price impact. The interface displayed warnings, but a checkbox confirmation was the only barrier. MEV bots extracted approximately $34 million via sandwich attacks. Contract-enforced guardrails — spending limits, minimum receive amounts, token whitelists — would have prevented this trade from executing.

3. Protocol Architecture

MindVault consists of six interconnected layers, each serving a specific function in the security and execution pipeline.

3.1 Client Layer

Users interact with MindVault through a frontend dApp (Next.js 14, wagmi v2, viem) or directly via smart contract calls. Wallet connection supports EIP-6963 compatible wallets (Rabby, MetaMask). AI trading agents connect via the EIP-712 signing interface, constructing and signing order intents off-chain. Supported LLMs include DeepSeek, Google Gemini Flash, and Kimi K2, optimized for cost-efficient trading cycles.

3.2 Guardrail Layer (AuthorizationModule)

The AuthorizationModule enforces per-agent constraints at the contract level. When a vault owner authorizes an agent, they configure four guardrails:

Token Whitelist: The agent can only trade tokens that appear on its approved list. Trades involving non-whitelisted tokens revert.

Daily Spending Limit: A hard cap on the USD value of trades the agent can execute per 24-hour period. Exceeding the limit reverts the transaction.

Role-Based Permissions: Agents are assigned roles with function-selector-level granularity. An agent authorized to call createOrder cannot call withdraw. Permissions are checked on every contract interaction.

Auto-Expiration: Every agent authorization has a timestamp after which it becomes invalid. Expired agents cannot submit orders. No manual revocation required.

Authorization requires three on-chain transactions: setUserRole (assign the agent a role within the organization), updateRole (grant the role permission to call specific function selectors), and setAgentGuardrails (configure the whitelist, spending limit, and expiration). This went through 6 contract iterations to harden the auth chain and has been tested end-to-end with live bot order creation.

3.3 Vault Layer (AssetManager)

The AssetManager is the core vault contract handling non-custodial fund management. It supports four asset types via the AssetType enum: Native (0), ERC20 Token (1), ERC721 NFT (2), and ERC1155 Multi-Token (3). Every asset is identified by an AssetKey struct containing the contract address, token ID, and asset type.

Key functions include deposit (with configurable fees via calculateDepositFee), withdraw (with configurable lock times via setWithdrawInfo and freeTimelock), internal account transfers (AccountTransfer), and order management (createOrder, createOrderBySig, fillOrder, cancelOrder).

Vault ownership is represented by an NFT minted through the ThreeMarketAccessPass contract via MembershipManager.enter(). Whoever holds the NFT controls the vault. This makes vaults transferable, composable, and governable. A single wallet can hold multiple NFTs and manage multiple vaults with different strategies.

3.4 Order Layer (EIP-712 Signed Intents)

Agents construct Order structs specifying: the asset to sell (AssetKey + amount), the asset to receive (AssetKey + minimum amount), an expiration timestamp, and fee details. The agent signs this structured data off-chain using EIP-712 typed signatures. The signed order is submitted via createOrderBySig, which verifies the signature using domainSeparatorV4 (tied to the specific chain and contract address, preventing cross-chain replay attacks), checks the agent's permissions via the AuthorizationModule, and posts the order on-chain.

The Order struct also supports an isAssetToReceiveUnknown flag for open RFQ where the vault doesn't specify the receive asset, enabling solvers to propose the best available fill.

3.5 Solver / Execution Layer (IOrderFiller)

External solver contracts implement the IOrderFiller interface and compete to fill posted orders. The fillOrder function on AssetManager accepts any contract implementing this interface, making the solver layer fully open and permissionless. The AssetManager atomically swaps: it sends the sale asset to the filler and credits the received asset to the vault's internal balance. If the filler doesn't meet the minimum receive amount, the transaction reverts.

Current and planned solver integrations include: Orbs Liquidity Hub (aggregated solver network with RFQ optimization), DEX routers (Uniswap, SushiSwap), Aave lending protocol (deposit/borrow through vaults), and any third-party solver that implements the IOrderFiller interface.

3.6 Blockchain Layer

MindVault deploys on any EVM-compatible chain. The entire frontend and contract config is driven by a single ACTIVE_CHAIN_ID variable. To deploy on a new chain, contracts are deployed and addresses updated in one config file. The EIP-712 domain separator includes the chain ID, preventing cross-chain signature replay. Currently deployed on Monad Testnet and Ethereum Sepolia, with SpookySwap live on Ethereum, Sonic, and GoatNetwork.

4. Governance

MindVault includes a ThreeMarketMultiGovernor contract with pluggable vote types and different rules per decision type. Vault NFT holders have voting power proportional to their holdings. The governor supports proposal creation, execution, and relay functions, with configurable voting delays and periods per proposal type. Vote managers can be registered per proposal type, enabling different governance rules for different categories of decisions.

5. Revenue Model

MindVault generates revenue through two primary channels:

Protocol fees: Approximately 5 basis points on order fills, configurable per organization via adminFeeOverride and updateDefaultDepositFee. Fee details are embedded in the Order struct's FeeDetails field, specifying the fee asset, amount, and the account where fees are credited.

Solver spread capture: The first-party solver ('Grim') captures spread between the order's minimum receive amount and the actual execution price when acting as a filler.

Secondary revenue streams under development include: hosted bot subscription service (pending regulatory clarity), premium marketplace features for strategy sellers, and advanced order type fees (dTWAP, dLIMIT via Orbs integration).

6. Agent Marketplace

MindVault will feature an open strategy/copy-trading marketplace where agent operators publish strategies with verified on-chain track records. Users subscribe to strategies by authorizing the operator's agent on their vault with their chosen guardrails. Fraud prevention mechanisms include: liquidity-aware execution (preventing front-running of large copy trades), seller staking requirements, tiered reputation systems based on verified performance, and on-chain track record verification that cannot be faked.

7. Integrations & Partnerships

Orbs: Early-stage integration discussions confirmed by Ran Hammer (VP of Business Development). Orbs' solver network would plug into MindVault's IOrderFiller interface, filling order flow using aggregated liquidity. Additionally, Orbs' dTWAP and dLIMIT protocols would provide advanced order types for MindVault users. Orbs is also developing MCP (Model Context Protocol) servers for their solver and perps infrastructure, enabling seamless integration with LLM-based agents.

SYMMIO: Perps integration under technical evaluation. SYMMIO's intent-based perpetual futures architecture aligns with MindVault's order model. MindVault would serve as the vault and guardrail layer in front of SYMMIO's settlement contracts.

SpookySwap V4: SpookySwap will be repositioned as an agentic DEX aggregator built on MindVault infrastructure, with 'Grim' as the default first-party trading bot.

8. Security Considerations

Smart contract risk: Mitigated by 6 contract iterations on testnet, non-custodial architecture limiting blast radius, and formal audit planned before mainnet deployment.

Agent key compromise: Attacker is bound by on-chain guardrails (whitelist, spending limit, expiration). Cannot withdraw funds or exceed configured limits.

Malicious solver: Orders enforce minimum receive amounts. Fill below the minimum reverts the transaction.

Oracle/price manipulation: Architecture uses solver competition for price discovery rather than oracle dependency, reducing this attack surface.

Cross-chain replay: EIP-712 domain separator includes chain ID and contract address. Signatures are chain-specific.

9. Technical Stack

Smart Contracts: Solidity, EIP-712 typed signatures, OpenZeppelin (AccessControl, ERC721Votes), deployed via Hardhat/Foundry.

Frontend: Next.js 14 (App Router), TypeScript, wagmi v2, viem, Tailwind CSS, React Query.

Indexing: Subgraph (Graph Node + Postgres + IPFS via Docker).

AI/Agent Layer: DeepSeek, Google Gemini Flash, Kimi K2 — evaluated for cost-efficient trading cycles. Each cycle requires 1-3 LLM calls. Costs are negligible at scale.

Chains: EVM-compatible. Currently Monad Testnet and Ethereum Sepolia. Deployable to any EVM chain without contract modifications.

10. Team

Owen Palmer — Co-founder, Product / BD / GTM. Core SpookySwap contributor since 2021. 5+ years in DeFi protocol development, product strategy, and business development.

Marko — Co-founder, Smart Contracts & Infrastructure. Smart contract architect who designed and built the full MindVault contract suite. Manages contract deployment, bot development, and backend infrastructure.

11. Roadmap

Months 1-3: Mainnet deployment, public vault launch, open SDK and IOrderFiller interface, strategy marketplace launch, first-party bot deployment.

Months 4-6: Ecosystem acceleration through chain hackathons and bounties, solver network expansion, perps integration.

Months 7-12: Scale to 20,000+ vaults, $25M+ TVL. Hosted bot subscription service, tiered marketplace with staking/reputation, advanced order types.