AI AGENTS × ON-CHAIN FINANCE

NINJA

Whitepaper — 2026-06 Edition

Agentic Security Controlplane for On-chain Finance & Agentic Finance

The Trust Layer between Intent & Execution — for the era when AI agents move money

> *What You Read Is What You Sign.*
>
> *Humans today. Agents tomorrow.*

ZKSC Inc.

2026-06-11

https://ninja.zksc.io · @NinjaScanBot (Telegram)

1. Abstract

AI agents have left the demo and entered real work. They write code (Agentic Coding), run support desks and back offices (Agentic Support & BPO), and detect and respond to threats (Agentic Security). And among the main use cases, the front line where agents touch money itself is **finance** — asset management, payments, trading. Humans today. Agents tomorrow. The migration from a present used by humans to a tomorrow used by agents has already begun.

When agents work in finance, what matters above all is **that the AI's behavior is visible**. What did the agent decide, why did it choose that trade, which risks is it taking — if this cannot be seen from the outside, delegation cannot hold, audits cannot be reproduced, and risk management cannot begin. The precondition for the era in which AI touches assets is not how smart the model is, but **the visibility and explainability of behavior**.

Today, the front line of this question is **on-chain finance** — finance that runs on public blockchains, commonly known as DeFi. The reason is structural: **ABC = AI × Blockchain × Coin**. Agents with intent run autonomously 24/7 (A); the blockchain gives those agents a shared ledger that never stops, resists tampering, and can be verified by anyone (B); and stablecoins enable programmable machine payments that do not depend on humans (C). Agents and on-chain systems are structurally made for each other, and the lead runner is Ethereum's on-chain finance. Total Value Locked (TVL) recovered to roughly $130 billion in early 2026 (second only to the 2021 peak in USD terms, and reportedly at an all-time high in ETH terms; source: DefiLlama).

So what is happening in that on-chain finance right now? Hacking losses reached the $3.4 billion scale in the most recent year (Chainalysis), and the center of gravity of fraud has moved from contract-code vulnerabilities to the **signing interface** — attackers moved to the signing screen, but the defenders have not moved yet. Blockchains were designed as "transparent finance where every transaction is public." In reality, bytecode is unreadable to humans, the internal allocations of Vaults are invisible, and calldata looks like nothing but strings of cryptographic data. **The financial system that proclaims transparency has become the most opaque financial system.** On top of this paradox, two facts converged in 2026. The Ethereum Foundation's formalization of Clear Signing (May 12) set in motion the standardization of making the signing target visible, and the rsETH incident of April (an unbacked mint of roughly $292 million) demonstrated that damage travels not through individual protocols (nodes) but **along structural dependencies between protocols (edges)** — reaching participants who never touched the asset in question. The further the standardization of "seeing" advances, the sharper the two blanks beyond it become: **visible but not readable**, and **hardening your own node does not stop the contagion**.

Ninja is the Agentic Security Controlplane that fills this blank — a **Trust Layer standing between Intent and Execution**. In the agentic-finance stack, standardization is moving ahead fastest in authentication & delegation (AP2 / ACP and peers) and in wallets & payments (x402 / MPP and peers). The third seat left in between — **monitoring & control** — is where Ninja stands. Through three Intelligence domains, Ninja Intelligence Core provides contract trustworthiness assessment (**NinjaCheck**), transaction-intent analysis (**NinjaScan**), and position-spillover forecasting on a cross-protocol dependency graph (**Ninja Position** — Systemic Position Forecast). All Intelligence follows a two-layer structure — deterministic Layer 1 (establishing facts) and LLM-augmented Layer 2 (translating facts into human language) — achieving reproducibility and explainability at once. As a slogan, we call the step beyond Clear Signing's WYSIWYS **What You Read Is What You Sign (WYRIWYS)** — only when "understand" is layered on top of "see" does it become "read." Only once you can read it should you sign it. As the layer above, we are designing ShoGun (Govern / Block), which carries Intelligence-driven policy control and automated action (not

yet shipped; phased introduction from Day 3 onward), aiming to run a single plane all the way from knowing to stopping.

The biggest update in this edition (2026-06) is that much of the text has shifted from "plans" to "where we are — implemented." NinjaScan / NinjaCheck are live on a Telegram Bot; the **MCP (Model Context Protocol) endpoint is already public, with no registration and anonymous access** (the API is offered B2B); and **pay-per-call billing over the x402 protocol is implemented** — an agent can call, and pay, with no human in between. Ninja Position runs as a PoC: for any wallet holding positions on supported protocols, entering a single address generates the dependency graph and replays incident contagion (coverage is expanding in stages).

The primary readers of this document are protocol operators, institutional investors, AI-agent developers, and on-chain finance users who need self-defense monitoring. This whitepaper describes Ninja's technical architecture; "not-so-smart-contracts," the database we built by analyzing more than 90 million EVM contract deployments; a case study of the rsETH incident; the product roadmap; the business model; and the competitive landscape.

2. Problem — The structural opacity of on-chain finance and Agentic Finance

The growth of on-chain finance is striking. TVL has recovered sharply, and emerging protocols such as Morpho have passed 1.4 million addresses (publicly reported). The DeFAI (DeFi + AI) sector is growing fast, and the age of Agentic Finance — AI agents trading autonomously on-chain — has arrived.

In the shadow of this growth, however, four structural problems remain unsolved.

2.1 Contract trustworthiness is unknown

When users access an on-chain finance protocol, their means of judging whether its contracts are safe are extremely limited. Is the contract verified? Does it use a proxy structure? Does it appear on the OFAC list? Do its bytecode characteristics resemble known exploit patterns? — expecting individuals to verify each of these one by one is not realistic.

2.2 "Seeing" alone does not protect a signature — three limits that remain after Clear Signing

Users cannot accurately understand what the transaction they are about to sign will actually do. And the center of gravity of fraud has moved precisely to this point. By industry analysis, roughly 80% of losses in 2022 stemmed from contract-code vulnerabilities; by 2024, interface- and signing-related causes accounted for the bulk of losses (about 86%), and the amount lost grew roughly 3.4x in two years ($0.7B → $2.4B). **Attackers moved to the signing screen. Defenders have not.** Against this "blind signing" problem, the Ethereum Foundation announced Clear Signing as an official activity on May 12, 2026 — a three-part standardization effort consisting of ERC-7730 (a structured display-descriptor format), a registry for sharing descriptors, and ERC-8176 attestations backing the accuracy of descriptors, aimed at displaying the signing target in a form humans can see. We welcome this. That pre-signature visibility is now treated as an industry-standard problem is genuine progress.

Picture the working reality, however — whales and operators facing tens to hundreds of signatures and multisig approvals a day — and three limits remain even after "seeing" is standardized.

Limit 1: Volume. Visible, but not readable to the end. No human can read a 100-page contract 100 times a day. As the signature count grows, review quietly hollows out. This is not negligence; it is the realistic constraint of attention. What is needed is not full-text display, but **showing, briefly and before signing, only the diffs and danger spots that are meaningful to that person**.

Limit 2: Where you should read depends on who is reading. In the same signature, what matters to a retail investor is the amount; to a whale, whether a permission carries an expiry; to a DAO signer, whether the counterparty contract is genuine. Web2 contract review works the same way: legal reads the termination clauses, accounting reads the payment terms, the business owner reads the counterparty and the amount. Whoever routes the approval has always marked it up — "look here." The same mechanism is needed in front of on-chain signing.

Limit 3: Clear Signing has structurally invisible territory. Clear Signing can display only those transactions for which a descriptor exists. A scammer does not ship their malicious contract with a label reading "this requests dangerous permissions." Without a descriptor, the wallet cannot translate the calldata, and we are back to blind

signing — **the most dangerous transactions carry no label.** Moreover, descriptors are in principle self-declared by application developers; false descriptors and registry poisoning are theoretically possible. What matters is that this is not a defect of Clear Signing. ERC-7730 states explicitly in its specification that it does not determine whether a transaction is safe. It is a designed non-goal. It follows that **safety can only be checked outside the standard, against sources independent of the descriptor.** This is not an option but a logical necessity — the same structure as a bank that, in collateral valuation, does not take the borrower's self-declaration on faith but cross-checks it against registries, appraisals, and market prices.

2.3 Structural contagion (edges) is no one's job

Tools that display wallet balances exist. But the problem lies beyond "what you hold." In the rsETH incident of April 2026 (detailed in § 6), withdrawals froze for users who had never touched rsETH or Aave at all — users who had simply deposited into Fluid Lite. The damage arrived along **structural dependencies that cross protocol walls (edges)** — collateral rules, shared Reserves, asset backing.

Each protocol can take responsibility only for what is inside its own boundary, and can see only the inside. Aave cannot see inside Fluid Lite; Fluid Lite cannot see the whole state of Aave's Reserve. The industry's post-incident measures — borrow caps, rate-limiting, isolated pools, and the rest — are all apt, but every one of them is about "how do I make my own node harder." **The cross-boundary edge itself falls, structurally and naturally, into no one's remit.** This blank is not anyone's negligence; it is a consequence of structure — and that is exactly why only a third-party layer can fill it.

2.4 AI-agent behavior cannot be monitored

The era in which AI Agents autonomously interact with on-chain finance protocols and execute trades has begun. Yet the delegating human has no way to know which contracts an Agent accesses, by what decision criteria it executes trades, or which risks it takes. A precursor already exists. In the Resolv incident of March 2026, a compromised private key was used to mint roughly 80 million USR without backing, causing a loss of about $25 million. The incident itself was not the work of an AI Agent. But the automated processing logic of surrounding protocols kept running after the depeg and amplified the secondary damage — the same shape as the failures that will multiply by orders of magnitude in Agentic Finance: **automated execution with no human judgment in the loop amplifies the damage.** In an era when autonomous Agents move funds at scale, incidents of this type grow in both quality and quantity.

Compounded, these problems sustain $3.4 billion in annual hacking losses, countless phishing scams, and a standing barrier to on-chain finance entry for institutional investors and AI-Agent operators.

3. Vision

Ninja's vision is to build the security and transparency Controlplane for on-chain finance and Agentic Finance.

3.1 Controlplane

"Controlplane" does not mean a tool that provides individual security functions; it means a foundation that controls security decisions, policies, and responses in an integrated way. Users and AI Agents alike can know the counterparty (Entity), read the action (Action), forecast asset spillover (Position), and on that basis define policies and execute automated actions through ShoGun.

3.2 What You Read Is What You Sign (WYRIWYS)

The cornerstone of Clear Signing is WYSIWYS — What You See Is What You Sign. We raise it one step. **Only when "understand" is layered on top of "see" does it become "read."** Next to the standardized display, an understanding layer is needed — one that carries the visualized information all the way to the point where that person, under that workload, can actually understand it. The three limits of §2.2 — volume, role, and invisible territory — each map to an element of Ninja's design: show briefly only the diffs and danger spots that matter; adapt the summary to the reader's role; and cross-check against sources independent of descriptors (on-chain facts and an analysis database spanning 90 million deployments).

3.3 The graph is most naturally drawn by a neutral third-party layer

A single protocol cannot easily draw the cross-protocol dependency graph to completion. Each protocol's remit and responsibility boundary close at its own node, and no structural incentive operates to maintain the full picture beyond its own wall. The work of connecting the scattered fragments into one sheet is most naturally carried by a **neutral third-party layer**. Just as Bloomberg once gave the market a shared information substrate and, in doing so, made every participant stronger, the on-chain finance dependency graph is not an enemy of protocols — it is a common substrate that pushes the ecosystem to its next stage. Ninja places this position — a neutral layer that sides with no particular protocol, wallet, or chain — at the center of its strategy.

3.4 Continuity from on-chain finance to Agentic Finance — a trust layer between Intent and Execution

On-chain finance has been kept at arm's length by regulators because of its "ambiguity of responsible parties." Paradoxically, the arrival of AI Agents gives this problem a thread toward resolution. Deterministic action logs, and the explainability grounded in the Layer 1 / Layer 2 two-layer structure, form a technical basis for proving "who made which decision, and why."

And in Agentic Finance the structure is the same as a human signature. Even if Clear Signing standardizes readability, without a layer for judgment and control an AI Agent is left in the state of "able to read, unable to stop." For an Agent that signs autonomously, that state simply means an incident. What Ninja aims to be is the **trust layer standing between Intent and Execution** — evaluate before execution (know), check against policy (govern), and stop it when necessary (block). In the agentic-finance stack, authentication & delegation (AP2 / ACP and peers) and

wallets & payments (x402 / MPP and peers) are being standardized first; the third seat left in between — monitoring & control — is where Ninja works.

4. Architecture — The overall structure of Ninja

Ninja's architecture consists of three layers.

```
Ninja (Agentic Security Controlplane)
│
├── Ninja Intelligence Core (Know)
│   ├── Entity Intelligence   – Know the counterparty (NinjaCheck)
│   ├── Action Intelligence   – Read the action (NinjaScan)
│   ├── Position Intelligence – Forecast the spillover (Ninja Position / SPF)
│   └── Learning              – Learn and evolve
│
├── Ninja Delivery Layer (Deliver)
│   ├── NinjaScan / NinjaCheck (Telegram Bot)
│   ├── Dashboard
│   ├── API
│   └── MCP + x402 (agent-native reference and billing rails)
│
└── ShoGun (Govern / Block) – phased introduction from Day 3 onward
    ├── Policy Engine
    ├── Whitelist Management
    ├── Approval Flows
    └── Automated Actions
```

4.1 The three layers

Ninja Intelligence Core (Know) is the intelligence engine at the heart of the Ninja platform. Through three Intelligence domains and Learning, it generates the Intelligence on which decisions are grounded. This separation lets analysis results be consumed consistently from every channel, and lets them serve as triggers for ShoGun's policy engine and automated actions. It provides the same Intelligence to AI-Agent calls as it does to human users.

Ninja Delivery Layer (Deliver) is the interface that delivers Intelligence to users and AI Agents. Today, the Telegram Bot (NinjaScan / NinjaCheck), the API, and the **MCP (Model Context Protocol) endpoint are live**. MCP is public with no registration and anonymous access, so AI agents and AI development environments can reference Ninja's Intelligence directly. The Dashboard is being implemented in stages, starting from risk comparison and user reporting.

ShoGun (Govern / Block) is the layer that defines and executes security policy based on Intelligence. **It has not shipped yet and will be introduced in stages from Day 3 onward** — a policy engine, whitelist management, approval flows, and automated actions (alert issuance, position-freeze proposals, and so on). By design it will later extend to defining behavior policies for AI Agents.

4.2 Product line

Product	Role	Status
NinjaScan	Transaction-intent analysis (`/scan`). The query window into Action Intelligence	Live
NinjaCheck	Contract trustworthiness assessment (`/check`). The query window into Entity Intelligence. **Wallet Address assessment to be added**	Live (extension planned)
Ninja Position (subtitle: Systemic Position Forecast, "SPF")	Forecasting and visualizing position spillover on a cross-protocol dependency graph	PoC live (one wallet address generates the graph; demo available)

4.3 Agent-native reference and billing rails — MCP + x402

The Delivery Layer's design principle is "the same Intelligence, for humans and for Agents, at minimum friction." Where we stand:

- **MCP (implemented):** the standard interface through which AI agents reference Ninja as a single source of truth (SSOT). Callable with no registration and anonymous access. Integration into your own systems and AI agents can start today
- **x402 pay-per-call (implemented):** support for the web-native metered-billing protocol built on HTTP 402 (Payment Required). With no account registration and no prior contract, an Agent pays per call and receives Ninja's Intelligence. It is the billing rail of the Agentic Finance era, in which AI agents purchase services autonomously — and a stepping stone toward Day 5 (the AI2AI Controlplane)

4.4 Risk Engine Marketplace — Intelligence, opened up

Ninja Intelligence Core is not designed as a single vendor's closed box. We are designing a framework (the Risk Engine Marketplace) that **accepts → evaluates → catalogs** third-party ML risk-assessment agents, and the catalog-browsing infrastructure is already running in production. The aim is a structure in which third-party detection engines that pass the quality gate can be added to the Intelligence Core. Combined with x402, the providers of assessment agents sell their assessments and callers pay per use — risk assessment itself becomes an agent economy. That is the direction in view.

5. Ninja Intelligence Core — 3 Intelligence + Layer 1 / Layer 2

5.1 Design principle: the two-layer architecture

All of Ninja's Intelligence strictly follows the two-layer structure below.

Layer 1 (deterministic layer)

- Establishes facts. 100% reproducible. Uses no LLM
- Always returns identical output for identical input
- Based on on-chain data, bytecode analysis, known-pattern matching, and the like
- The output of this layer is the foundation of every decision

Layer 2 (LLM-augmented layer)

- Translates Layer 1 output into a form humans understand easily
- Supplements and explains facts, but does not override Layer 1 determinations
- What the LLM generates is "explanation," not "judgment"

This design exists to secure reproducibility and explainability — the properties that matter most in a security product. LLM output is probabilistic and can return different results for identical input. Entrusting the basis of security judgment to an LLM not only causes misjudgment; it makes the grounds of a determination hard to reproduce in audits and incident response. Layer 1 fixes the facts; Layer 2 only renders those facts in a readable form — this principle is the foundation of Ninja's trustworthiness.

This two-layer structure is also the implementation principle of WYRIWYS (§3.2). Showing briefly "the spots this person should look at now" is Layer 2's job — but **the retrieval of facts is never delegated to a probabilistic model**. Who, how much, which permissions, how similar to past behavior — the facts are obtained deterministically, and role-aware summarization is layered on top. The same holds for monitoring AI-Agent behavior. If an Agent's trades are merely "interpreted" by an LLM after the fact, the interpretation itself becomes unreproducible. Layer 1 fixes the factual structure of the trade, and Layer 2 renders it in an auditable form — this structure is the technical foundation on which Agent accountability is implemented.

5.2 Trust & Reliability — is the product that protects you itself protected?

A security product is itself an attack target. Ninja runs internal security reviews of its own platform aligned with the OWASP framework, and has systematically resolved the findings across the major domains — from authentication, session management, and input validation through supply-chain hygiene — as a hardening program (as of 2026-06).

The treatment of the LLM layer deserves particular note. Because Layer 2 uses LLMs, prompt injection can become an attack surface of Ninja itself. Ninja **operates an evaluation infrastructure for prompt-injection resistance (continuous evaluation against a corpus of injection patterns systematized by path × technique)**, iterating between defense implementation and evaluation. This is a direct response to the risk that the OWASP Top 10 for LLM Applications lists first (LLM01: Prompt Injection), and it supports, on the operational side, the Layer 1 / Layer 2 principle of never delegating judgment to the LLM.

(Individual vulnerability details and inspection results are omitted from this document for security reasons. We respond to institutional-investor and partner due diligence individually under NDA.)

5.3 Entity Intelligence — Know the counterparty (NinjaCheck)

Entity Intelligence assesses the trustworthiness of contracts and addresses from multiple angles. Its query window is **NinjaCheck** (live as the Telegram Bot's `/check`).

Layer 1 evaluation items:

- Bytecode ML analysis: extracts features from contract bytecode and detects malicious patterns with machine-learning models
- Risk Score computation: scoring that integrates multiple evaluation axes with weighting
- Blacklist / OFAC matching: cross-checks against sanctions lists and known malicious addresses
- Contract attribute verification: verified status, proxy structure, time elapsed since deployment, and so on

Layer 2 augmentation:

- Natural-language summaries of assessment results
- Prioritized explanation of risk factors
- Comparative context against similar contracts

The ML risk engines behind the assessment are integrated in production as a fan-out across multiple specialized agents (malicious-contract detection, Ponzi-style protocol detection, and others).

Planned extension: today's NinjaCheck targets contract addresses. The next extension adds **Wallet Address assessment** — evaluating a counterparty wallet's history, associations, and risk profile. Beyond that, the role-aware summaries described in § 3.2 — "in this signature, here is what you should look at now" — will be integrated into the NinjaCheck query experience.

Operating modes:

- **On-demand:** instant assessment initiated by the user via the `/check` command
- **Continuous:** automatic re-assessment when changes (upgrades, owner changes, and so on) are detected in dependency contracts

5.4 Action Intelligence — Read the action (NinjaScan)

Action Intelligence analyzes a transaction's calldata and determines what the transaction is trying to do. It processes not only transactions that humans sign but also transactions that AI Agents autonomously generate and execute, through the same analysis pipeline. Its query window is **NinjaScan** (`/scan`).

Layer 1 analysis pipeline (Action Intelligence Pipeline):

```
ParseCheck
  → TxIntentMapper (intent classification)
    → ContractArtifactResolver (contract information resolution)
      → Parallel Checks (multiple verifications run concurrently)
        → Layer 1 Output (structured facts)
          → LLM Augmentation (Layer 2)
```

1. **ParseCheck:** syntactic parsing and basic validation of transaction data
2. **TxIntentMapper:** mapping from calldata function selectors to intent categories (13 types)
3. **ContractArtifactResolver:** resolution of the target contract's ABI, source code, and metadata
4. **Parallel Checks:** concurrent execution of multiple verification logics (approval-amount validation, known-exploit-signature matching, reentrancy checks, and others)
5. **Layer 1 Output:** structured output of category (13 types), alert codes (17 types), and the traffic light (SAFE / WARNING / DANGER)
6. **LLM Augmentation:** turns the Layer 1 structured output into a human-readable analysis report

The 13 categories: Native Transfer, Token Transfer, Token Approve, Token Approve All, Permit, Permit2, DeFi Swap, Multicall, Multisig Exec, Ownership Transfer, Proxy Upgrade, Generic Call, Unknown

The traffic-light system:

- **SAFE:** matches known patterns; no anomalies
- **WARNING:** contains elements requiring attention (large approvals, unverified contracts, and so on)
- **DANGER:** high-risk elements detected (known exploit signatures, association with sanctioned addresses, and so on)

Analyzing multi-layer transactions: real attacks often hide inside transactions nested in multiple layers. Informed by the Safe (multisig) `execTransaction`-style attack used in the 2025 Bybit incident, NinjaScan decodes the target inside the multisig execution and brings new contracts and dangerous calls lurking inside into detection scope (support for deeper multi-layer nesting is expanding).

Multichain: the analysis pipeline is chain-neutral by design, and coverage is expanding beyond Ethereum to emerging chains in sequence (including automatic chainId detection from raw transactions and Proxy resolution across verification explorers).

Complementarity with ERC-7730: as Clear Signing descriptors spread, Action Intelligence's display context grows even richer. At the same time, it is precisely on descriptor-less and spoofed transactions (§2.2, Limit 3) that NinjaScan's cross-checks bite — grounded in on-chain facts and the analysis database, independent of descriptors. The standardized "see" and the independent-source "understand" do not compete; they are clearly complementary.

Operating modes:

- **On-demand:** instant analysis initiated by the user via the `/scan` command
- **Continuous:** automatic detection and analysis of transactions from monitored wallets (including AI-Agent wallets)

5.5 Position Intelligence — Forecast the spillover (Ninja Position / SPF)

Position Intelligence is the domain that has evolved most in this edition. The product is **Ninja Position**, subtitled Systemic Position Forecast (SPF) — not "a list of your positions," but a forecast, **on the cross-protocol dependency graph, of what reaches you next**. Note that "forecast" in this document means reachability over the dependency structure — showing in advance through which paths an impact can arrive at you — not a probabilistic prediction of prices or timing.

5.5.1 The 4-edge model — damage travels along edges, not nodes

What a wallet holds (the active edge) is visible to anyone; call `positionsOf(actor)` and you have it. That is not where the difficulty lies. What actually carries the damage are **three kinds of latent edges that bind you structurally even though no money moves**.

Edge	What it is	Example
active edge (visible)	actual holding / supply / borrow positions	Vault shares; Aave supply / borrow
CollateralEdge (latent)	collateral rules: this asset can be posted to borrow that one	rsETH can collateralize a WETH borrow
shared Reserve (latent)	shared markets: multiple actors using the same Reserve	the same Aave WETH Reserve
Asset.backing (latent)	the asset's backing chain	rsETH → stETH → ETH

The rsETH cascade (§ 6) can be reconstructed completely as a single traversal across these four edge types.

5.5.2 Hub reverse-lookup — risk you cannot reach by walking from yourself

The query that bites is the **reverse-lookup** with a shared Reserve as the hub. Enumerate, in reverse, every actor holding a position on a given Reserve, and the relationship "you and the attacker are connected through the same WETH Reserve" becomes visible. This is a risk that naively walking `positionsOf(0xYou)` from your own wallet can never reach, in principle.

That said, the reverse-lookup operation itself is something anyone can write once the structure is visible — it is not hidden magic. The hard part is not the edge-drawing primitive; it is **connecting the fragments scattered across protocol walls into a single graph**, and the most natural owner of that work is a neutral third-party layer (§ 3.3).

5.5.3 One graph, three readouts

Once the graph can be drawn as a single sheet, the use cases follow from it as readouts.

- **Actor foresight:** query the graph from one node — "your next move at T+1 minute." Self-defense monitoring for individuals, whales, and operators
- **Protocol systemic risk:** aggregate the graph — on which edges dependency concentrates, and how far stress conducts
- **Incident Overlay (time-series replay):** replay the graph over time — along which path, to which actors, in which order an incident arrived

5.5.4 Where we are, and an honest scope

Ninja Position runs as a PoC. Enter one wallet address, and it reads Vault shares, lending and borrowing positions, collateral rules, and asset backing from on-chain state, generating the structural graph dynamically. In the Incident Overlay built on the rsETH incident, the phases — normal state (T=-1), anomalous-mint detection (T=0), the attacker's supply (T=1), and contagion through the shared Reserve (T=2) — are layered onto the structural graph in time series, and **the impact path can be tracked on screen until it reaches YOU**. Supported protocols are expanding in stages from the major lending family (Aave, Fluid, Spark, and others) and LST/LRT backing relationships. Some collateral and backing relationships are based on verified static definitions today; over the medium term these migrate to automatic updates driven by on-chain monitoring.

At the same time, we want to be honest about scope. **Upstream auto-detection — a mainnet subscriber that catches anomalous mints in real time — Ninja Position does not build, and will not own.** That is the domain of existing, excellent services (Hypernative, Forta, Blockaid, Gauntlet, Chaos Labs, and others). What Ninja Position concentrates on is **the edge-drawing layer — the cross-protocol dependency graph itself**. Visualization is not the endpoint. Only once you can see do the next things begin — simulating in advance, designing exit routes, quantifying risk, provisioning for a crisis with a basis — and beyond that opens exactly the domain where Programmable Finance, in which conditions are composed in code and the result is verifiable by anyone, is at its best.

5.6 Learning — Learn and evolve

Ninja evolves continuously through three mechanisms.

1. **User reports → Ground Truth → model retraining:** feedback from users (false-positive reports, suspicious-transaction reports, and so on) is accumulated as Ground Truth and used to improve detection-model accuracy. The reporting UI is implemented, and an evaluation database for systematically verifying judgment accuracy is being built out
2. **Alert-code accumulation:** each time a new incident occurs, its pattern is analyzed and added as a new detection rule (alert code). Detection coverage keeps expanding over time (threat-database build-out is ongoing)
3. **External knowledge absorption:** continuous ingestion from external sources such as public threat intelligence and community data (via public feeds and official APIs) keeps Ninja's knowledge base updated

6. Case Study — The rsETH Cascade (April 2026)

This chapter reconstructs the rsETH incident — in reach, the defining systemic incident of the first half of 2026 — through the lens of the dependency graph. The analysis in this chapter is based on our reply (Medium, 2026-06-01) to *Kelp Incident Review*, the post-mortem Lido published on May 21, 2026.

6.1 What happened

On April 18, 2026, the LayerZero bridge securing KelpDAO's rsETH was compromised (involvement of the Lazarus Group has been indicated), and **roughly 116,500 rsETH (about $292 million) was minted with no backing behind it**. Five weeks on, the incident was officially "resolved." KelpDAO restored its backing ratio to 100.01%, redemptions normalized, and the lockbox was re-secured by dual verification from LayerZero and Chainlink. Lido's EarnETH vault reopened on May 15, and the final loss of 143.98 ETH was covered in full by the DAO's first-loss layer. Aave's DAO voted to liquidate the attacker's frozen funds.

But what each protocol repaired was its own **node**. The **edge** that carried the damage — the dependency graph spanning protocols — still does not exist in a form anyone can consult as a public good.

(The narrative and figures in this chapter are based on Lido's *Kelp Incident Review* (2026-05-21) and public governance records, as of 2026-06-01.)

6.2 The cascade as structure — one 4-edge traversal reproduces it

Rewrite the incident as structure, and it comes out in four steps.

1. The attacker minted rsETH with no backing → rsETH's backing-asset chain (**Asset.backing**) broke
2. Aave had a rule that "rsETH can be posted as collateral to borrow WETH" → that is the **CollateralEdge**
3. As a result, Aave's WETH Reserve pinned at 99% utilization, and funding stress propagated to every actor sharing that same WETH Reserve (the **shared Reserve** edge)
4. And **withdrawals froze for people who had never once touched rsETH**

Trace that last line concretely. There was a user who had simply deposited into a Fluid Lite vault. They held no rsETH. They had never opened a position on Aave. All they held was a Fluid Lite vault share. But Fluid Lite, to run that share, was borrowing WETH from the shared Aave WETH Reserve to turn an stETH loop. The moment the rsETH incident pinned that WETH Reserve at 99%, there was physically no WETH left to withdraw; Fluid Lite could not unwind the loop, and the depositor's withdrawal froze.

This was not unique to Fluid Lite. rsETH-linked positions across several major lending markets, and the many loop positions that depended on the shared WETH Reserve, hit freezes and failed unwinds one after another. **From the attacker to you, money never moved directly even once. And yet the damage arrives — because damage travels along edges (structural dependencies), not nodes (individual positions).**

6.3 Reproduction in Ninja Position — the Incident Overlay

Ninja Position's PoC reproduces this cascade as a single traversal of the 4-edge model, and it has been tested. The Incident Overlay replays the incident in four phases on the time axis.

Phase	What happens on the graph
T=-1 (normal)	the structural graph: your holdings, Fluid Lite's Aave position, Aave's collateral rules, and asset backing are visible statically
T=0 (detection)	the anomalous rsETH mint is detected as the root event; rsETH and the related Reserves are highlighted as watch targets
T=1 (supply)	the attacker supplies rsETH to Aave; the attacker–rsETH–Aave rsETH Reserve relationships overlay as temporal edges
T=2 (contagion)	the impact path through the shared WETH Reserve is drawn, **reaching YOU (your position) as the endpoint of the path**

This is not a paper diagram. For any wallet holding positions on supported protocols (currently Aave / Fluid / Spark, with more coming), entering a single address makes your own version of this graph run on the PoC.

6.4 The protocol-side response, and the blank that remains

Lido's review is a serious, industry-facing proposal presenting seven-plus framework changes. On Aave's governance forum, too, debate ran active — more than 150 posts by over 60 participants (as of 2026-06-01). The measures raised — borrow caps, rate-limiting, cooldowns, isolated-pool design, reputation-tiered borrowing, direct supply-side monitoring — are all apt and necessary. But the vantage point they share is "how do I make my own protocol (node) harder."

What is interesting is that when you read Lido's change items, most of the major changes map naturally onto "edges" of the dependency graph.

Lido framework change	The graph edge it maps to
① Direct supply-side monitoring	Asset.backing edge (watching the backing degrade)
② Second-order effects as first-class	shared Reserve edge (spillover on a shared Reserve)
⑤ Deeper recursive-strategy review	path simulation along edges
⑥ Decoupling yield from exit quality	node stress scoring (high APY ≠ safe)

The protocol side, too, is arriving independently at the same structure. This is less the arrival of a competitor than the best evidence that this category is real. But each protocol can draw only the edges within its own remit, and the fragments do not naturally connect into a single sheet. The most natural owner of the one graph is a neutral third-party layer (§ 3.3) — and that is where Ninja Position stands.

7. not-so-smart-contracts — A large-scale EVM contract analysis database

Ninja maintains a proprietary large-scale EVM contract analysis database, "not-so-smart-contracts." Concretely, across EVM-compatible chains it has **analyzed more than 15 million unique bytecodes deduplicated from over 90 million contract deployments** (the actual analysis count after consolidating re-deployments of identical bytecode). The database is named in homage to the smart-contract vulnerability pattern collection of the same name published by Trail of Bits (Crytic) (https://github.com/crytic/not-so-smart-contracts). Where the Trail of Bits repository is an educational resource for studying patterns, Ninja's not-so-smart-contracts is a production database built by analyzing EVM contracts at scale — different in both purpose and magnitude. This database strengthens Ninja Intelligence Core in two directions.

7.1 Strengthening Action Intelligence: selector mapping

From the function-selector-to-method-name mappings extracted from the analyzed contract corpus, those of high validity are selected and integrated into Action Intelligence's intent classification (**live in production**). This resolves method names broadly from function selectors even for transactions against unverified contracts whose ABIs are not public. Intent classification of transactions previously judged "Unknown" has improved substantially, expanding Action Intelligence's coverage.

7.2 Strengthening Entity Intelligence: bytecode embeddings

Features are extracted from each contract's bytecode and stored as vector embeddings. This allows the bytecode of a new contract to be compared for similarity against the corpus of known contracts. Judgments such as "high structural similarity to the corpus of past exploit-target contracts" become possible, giving Entity Intelligence's risk assessment quantitative grounding. Even for unknown contracts, risk can be detected early through structural similarity to past exploit patterns.

7.3 What it means as an independent source

As stated in §2.2, Clear Signing descriptors are developer self-declarations. This database, spanning 90 million deployments, underpins Ninja's cross-checking capability as an **independent source** that does not rely on self-declaration. For contracts with no descriptor — or with spoofed ones — an assessment can still be assembled from the on-chain facts: bytecode and history.

8. Product & Customer Roadmap — Day 1 to Day 5, and where we are

Ninja's product expands in five stages, from Day 1 to Day 5. Each Day progressively extends target customers, channels, Intelligence domains, operating modes, and pricing models.

8.1 Where we are (June 2026)

Before describing the roadmap, here is what is implemented and running at this point.

Area	Live (2026-06)
Bot	NinjaScan `/scan` (TX analysis), NinjaCheck `/check` (contract assessment) — publicly launched
API / MCP	B2B API infrastructure; **MCP endpoint (no registration, anonymous access)**
Billing rail	**x402 pay-per-call (implemented)**
Intelligence	ML risk engines integrated in production (fan-out), selector DB integration, recursive decoding of Safe `execTransaction`, expanding multichain support
Position	**Ninja Position PoC** (dependency-graph generation + rsETH Incident Overlay; demo available)
Dashboard	phased implementation (risk comparison, user-report UI)
Operations	usage-metrics measurement, operations dashboard, security hardening (§ 5.2)

MCP support, originally placed at Day 3, was shipped ahead of schedule in response to reference demand from AI agents. x402 was an item on which no adoption decision had been made when the roadmap was drawn; it was implemented additionally as Agentic Finance advanced.

8.2 Day 1: Foundation — B2C customer base acquisition [current]

- **Target customers:** B2C individuals
- **Channels:** NinjaScan / NinjaCheck (Telegram Bot) + MCP (AI agents)
- **Intelligence:** Entity Intelligence + Action Intelligence
- **Operating mode:** On-demand (user-initiated)
- **Pricing:** free by default (Free) + metered access via x402. Day 1's primary objective is customer-base acquisition; subscription billing begins in earnest from the next Day
- **In parallel:** secure one or two B2B pilot customers as advance preparation for B2B expansion from Day 2 onward. Add **NinjaCheck Wallet Address assessment** as a near-term extension

8.3 Day 2: Continuous Monitoring — B2C passive monitoring + monetization launch

- **Target customers:** B2C individuals (continued)
- **Channel added:** Dashboard (general availability)
- **Intelligence added:** Position Intelligence (Ninja Position PoC → productization) + early Learning (report-data integration)
- **Operating mode added:** Continuous monitoring (passive alerts)
- **Pricing:** launches in earnest. Three plans: Free / Lite / Pro

- **Prerequisite:** willingness-to-pay validation from Day 1 is complete, and B2B pilots are proceeding in parallel

8.4 Day 3: Protocol Expansion — B2B API + early ShoGun

- **Target customers:** on-chain finance protocol operators
- **Channel added:** full rollout of the B2B API (three-tier structure: Read / Monitor / Control APIs)
- **Features added:**
  - protocol-specific monitoring items (flash-loan countermeasures, Vault strategy-deviation detection, governance-proposal analysis)
  - **Ninja Position readouts for protocols** (systemic-risk aggregation, § 5.5.3)
  - early ShoGun (Govern / Block): policy engine, whitelists, approval flows
- **Intelligence:** Learning in full operation

8.5 Day 4: Institutional Control — ShoGun expansion

- **Target customers:** CEXs, institutional investors, family offices, compliance teams
- **Features added:** custom governance-policy DSL, expanded automated actions, automated generation of compliance reports
- **Channels:** dedicated institutional Dashboard, SLA-backed API line

8.6 Day 5: AI↔AI Controlplane

- **Target customers:** AI-agent providers, the AI2AI ecosystem
- **Features:** one step beyond MCP support (AI → Ninja reference), providing **policy mediation, contract verification, and incident isolation between AI agents**
  - inter-Agent transaction-policy agreement protocol
  - cryptographic proof of action logs
  - automatic isolation of anomalous Agents
- **Positioning:** the completed form of trust infrastructure for the Agentic Finance era. **The already-implemented MCP (reference rail) and x402 (payment rail) are the foundation of this AI2AI Controlplane**

Integrated roadmap table

Phase	Primary customers	Channels	Intelligence	Operating mode	Pricing
Day 1 Foundation [current]	B2C individuals + AI Agents	NinjaScan / NinjaCheck + MCP	Entity + Action (+ Position PoC)	On-demand	Free by default + x402 metered
Day 2 Continuous	B2C individuals	+ Dashboard	+ Position productized + early Learning	+ Continuous	Free / Lite / Pro
Day 3 Protocol	On-chain finance protocols	+ full B2B API	+ full Learning	+ early ShoGun	B2B API pricing
Day 4 Institutional	CEXs / institutions	+ dedicated Dashboard	(same as above)	+ ShoGun expansion (governance DSL)	Enterprise pricing
Day 5 AI2AI	AI-Agent providers	+ inter-Agent protocol	(same as above)	+ AI2AI mediation	Agent pricing (x402-based)

9. Competitive Landscape

Multiple incumbent players operate across on-chain finance security, portfolio visualization, and Agentic Finance security. This chapter classifies them into three archetypes by the nature of their capabilities and makes the differences from Ninja visible.

9.1 Archetype definitions and the capability matrix

- **Security Detection:** detection and notification of danger at the core, in some cases extending to execution blocking. Blockaid (Cosigner), GoPlus, Hypernative (Firewall), Forta, and others
- **Asset Visualization:** specialized in position display, without providing security assessment. Zapper, Zerion, DeBank, Exponential.fi, and others
- **Control Layer:** extends to integrated, policy-based execution control. DeFi Saver (specialized in automated execution; partial) and **Ninja** (ShoGun in phased introduction)

Note: this table is our independent assessment, based on public information as of June 2026, of the major players with meaningful funding and adoption. For agent-native newcomers (Openfort, Human.tech, and others), see the caveats in § 9.5.

Archetype	Service	Entity	Action	Position	LLM explanation	Alerts	Dashboard	Execution control	B2B API	AI-facing
Security Detection	Blockaid	△	○	x	△	○	△	○ (Cosigner)	○	△
Security Detection	GoPlus	○	△	x	x	△	x	x	○	○
Security Detection	Hypernative	△	○	x	x	○	○	○ (Firewall)	○	x
Security Detection	Forta	△	○	x	x	○	△	△	○	x
Asset Viz	Zapper	x	x	△	x	x	○	x	△	x
Asset Viz	Zerion	x	x	△	x	x	○	x	x	x
Asset Viz	DeBank	x	x	△	x	△	○	x	△	x
Asset Viz	Exponential.fi	x	x	○	△	x	○	x	x	x
Control Layer	DeFi Saver	x	x	△	x	○	○	○ (auto-execution)	x	x
Control Layer	**Ninja**	○	○	○	○	○	△ (phased implementation)	Planned (ShoGun)	○	○ (MCP / x402)

○ = offered as a core capability △ = partially supported x = not supported (our independent assessment based on public information as of June 2026)

The **Security Detection archetype** excels at "alerting to danger," and some players — Blockaid's Cosigner, Hypernative's Firewall — extend to blocking execution of individual transactions. The old dividing line, "incumbents

only ring alarms," is no longer accurate. At the same time, no vendor in this table offers the cross-protocol dependency graph (Position) as a core capability. The **Asset Visualization archetype** excels at visualizing "what you hold" but provides no security assessment. In the **Control Layer archetype**, vendors that integrate detection, visualization, explanation, policy control, and AI-facing rails on a single plane remain few, and the AI-facing Control Layer is, at this point, an emerging segment still in demand validation. Ninja's differentiation lies not in the presence or absence of individual features but in this **integration onto a single plane** and in **the dependency graph as its held ground**.

9.2 Ring the alarm on the node, or draw the edge

The difference in the Position domain is not only a matter of capability but of **vantage point**. Protocol-level anomaly detection (Forta, Hypernative, and others) rings an alarm on a node — an individual protocol or contract. That is necessary and excellent work. What Ninja Position draws is the **edge** between nodes, translating "an anomaly that happened elsewhere" all the way to "and along this structural dependency it reaches you." The two are different posts on the same question; as stated in § 5.5.4, upstream detection is not Ninja's domain — and edge translation is not the domain of the existing detection services.

9.3 The blank market — four quadrants of "to whom" × "what"

Lay out the market for risk information along "to whom it is delivered (Protocol ↔ Customer)" × "what is delivered (Alert ↔ Recommendation)," and a blank appears.

	Alert (inform)	**Recommendation (down to how to act)**
For protocols	Forta, Hypernative, etc.	Gauntlet, Chaos Labs, etc.
For customers (depositors, whales, operators)	DeBank, Zapper, Nansen, etc.	**(no major player as of 2026-06) ← Ninja Position**

Protocols receive both alerts and risk advisory. Customers — the side entrusting their assets — receive alerts at most; in the **Customer × Recommendation** quadrant — "your position is exposed along this path, so here are your options for how to move" — no major player is visible as of June 2026. Ninja Position stands in that quadrant.

Note: "Recommendation" in this table means the presentation of position-specific options for action, grounded in forecasts of cross-protocol contagion paths. DeFi Saver's automated execution (automation of a single position) and Exponential.fi's static risk ratings fall outside this definition. The protocol-facing readout (§ 5.5.3) does not compete with the economic simulation of Gauntlet and peers; it is a complementary approach that starts from the structural graph.

9.4 Relationship to the Clear Signing ecosystem — complementary, not competitive

The Ethereum Foundation's Clear Signing (ERC-7730 / registry / ERC-8176) is an activity that standardizes making the signing target visible; it is not a competitor to Ninja. Rather, the further standardization advances, the sharper the need becomes for an independent assessment layer that fills its designed non-goal — judging safety (§ 2.2). In a world of well-maintained descriptors, Ninja's Action Intelligence gains richer display context; in the descriptor-less and spoofed territory, its value as an independent cross-checking source stands out. The two are clearly complementary.

9.5 Ninja's differentiation points, and caveats

- A design that **integrates onto a single plane**: detection (Entity / Action), visualization (Position), explanation (LLM), control (ShoGun, phased introduction), and AI-facing rails (MCP / x402)
- Reproducibility and explainability through the two-layer architecture of deterministic Layer 1 and LLM-augmented Layer 2
- **The cross-protocol dependency graph (Ninja Position / SPF) — the held ground that a neutral third-party layer occupies most naturally**
- Agent-facing reference (MCP) and billing (x402) rails **already implemented**, with the roadmap to Day 5 (AI2AI) stated explicitly

Caveat: AI-facing security is not a "blank" — it is a space **filling rapidly**. Alongside incumbents' AI support (GoPlus's AI Agent Security API and others), agent-native newcomers (Openfort, Human.tech, and others) have also appeared. Ninja does not assume the blank will persist; its strategy is to keep the lead through the combination of dependency graph × neutral layer × single-plane integration. The Customer × Recommendation market, too, is at the demand-validation stage; we expand in stages — B2C → B2B protocols → AI-facing — and make each next investment decision on the validation results of the preceding phase (competitive assessments are our own, based on public information as of June 2026).

10. Business Model

Ninja's business model consists of three pillars: consumer freemium, B2B API, and agent-facing metered billing (x402).

B2C (individuals)

Three plans: Free / Lite / Pro. Higher plans expand usage limits and features.

Plan	Details
Free	`/check` 3 uses/day + `/scan` 3 uses/day (as of 2026-06)
Lite	`/check` expanded daily uses + `/scan` expanded daily uses
Pro	Unlimited + priority response + detailed report output

Day 1 concentrates on customer-base acquisition with the basic free plan; Lite/Pro launch in earnest from Day 2 to begin monetization. The Free tier continues to function as a user-acquisition channel beyond Day 1.

B2B API

Three tiers: Read API / Monitor API / Control API. Higher tiers expand the functional scope, with pricing stepping up accordingly. Prices are decided individually through agreement with pilot customers, referencing benchmarks based on public information and market interviews.

Plan	Capabilities	Target categories (examples)
Read API	Entity / Action Intelligence reference	Curators, wallet providers
Monitor API	+ Continuous monitoring + alert delivery + protocol-specific monitoring items	On-chain finance protocols, AI-Agent providers
Control API	+ ShoGun policy engine + automated actions + governance DSL	Institutional investors, compliance, CEXs

Agent-facing metered billing — x402 pay-per-call (implemented)

As the third rail, **pay-per-call billing over the x402 protocol is implemented**. With no account registration, no prior contract, and no API-key issuance, an AI agent (or a human) completes payment at the HTTP layer on each call and obtains the Intelligence. It monetizes exactly the use that fits neither subscriptions nor enterprise contracts — one-off use by autonomously operating Agents. It is the Agentic Finance-native billing model, and the foundation for Day 5 Agent billing.

Note: enterprise full-suite offerings (MCP integration, SLA guarantees, dedicated support) are handled individually.

11. Team

Core team

- **CEO Kaneshiro:** 13 years at Accenture. Led numerous AI projects for financial institutions. End-to-end experience from strategy formulation to technology delivery
- **CTO Murakami:** Technical leader in large-scale system development. Deep expertise in distributed system design and performance engineering
- **Megami:** Hands-on experience at a crypto exchange and at Fireblocks (institutional crypto infrastructure). Practical knowledge of security operations. Leads development of the Ninja Position (SPF) PoC
- **Kai Otsuki:** General Co-Chair, B4TI (IEEE International Workshop on Blockchain for Decentralized Trust and Digital Identity). Contributor to the Ethereum Foundation and participant in IETF SD-JWT (RFC 9901) standardization
- **CMO Suganuma:** Trilingual (Japanese, English, Chinese). Global marketing and community building

Advisors

- An ICANN DNSSEC key manager (Trusted Community Representative) — one of a small number of security experts worldwide who take part in the most critical cryptographic key-management ceremonies of internet infrastructure (name withheld)

12. Market Environment and Regulatory Trends

Market data

- **On-chain finance TVL:** over $130B (early 2026; second only to the 2021 peak in USD terms, and reportedly at an all-time high in ETH terms; source: DefiLlama)
- **Web3 security market:** $2.9B (2025) → $15.8B (2032 projection, CAGR 26.4%)
- **Annual hacking losses:** $3.4B (2025; source: Chainalysis)
- **rsETH incident:** April 2026. A LayerZero bridge compromise minted roughly 116,500 rsETH (about $292M) with no backing. Contagion reached the users of multiple protocols through a shared Reserve — the defining systemic-cascade incident (§ 6)
- **Resolv incident:** March 2026. A compromised private key minted roughly 80 million USR without backing; a $25M loss. The automated processing logic of surrounding protocols kept operating after the depeg and amplified secondary damage — making the security risk of Agentic Finance concrete
- **Morpho:** over 1.4 million addresses (publicly reported). Emblematic of the rapid expansion of Vault-based on-chain finance
- **DeFAI (DeFi + AI):** a rapidly growing sector. On-chain trading by AI agents is forming a new market category

The nature of on-chain finance risk changes with the era. After the single-protocol incident period (2021–22), the bridge incident period (2023), and the LRT-nesting period (2024–25), the first half of 2026 entered the **systemic-cascade period, in which a single incident chains across protocol walls**. The rsETH incident is its archetype, and this document's argument — from node-level defense to edge-level visibility — answers this shift in the market environment.

Regulatory and standardization trends

Ethereum Foundation formalizes Clear Signing (May 12, 2026)

The EF announced Clear Signing as an official activity. The structure: ERC-7730 (the structured-data Clear Signing format; updated to V2 in April 2026, Draft status) as the descriptor format; a registry hosted by the EF as a neutral steward for sharing descriptors; and ERC-8176 attestations backing their accuracy. The EF itself states explicitly that registry inclusion implies no audit or guarantee of safety — judging safety is left outside the standard, to independent assessment layers. As argued in this document (§ 2.2, § 9.4), this is a tailwind that underwrites the market need for Ninja.

SEC no-action position on DeFi front-ends (April 13, 2026)

The U.S. SEC's Division of Trading and Markets issued a staff statement (a no-action position) to the effect that it would not require broker-dealer registration of DeFi front-end providers meeting certain conditions. The main conditions are (1) user autonomy, (2) neutrality, and (3) transparency (disclosure of conflicts of interest, cybersecurity measures, publication of MEV strategies); the measure is time-limited to five years (through April 13, 2031). The third condition in particular — the transparency requirement — aligns in direction with the value Ninja Intelligence Core provides. Ninja's B2B API can support front-end providers in the practical work of satisfying the conditions (cybersecurity measures and assured transparency).

Other regulatory developments

- **MiCA (EU):** the comprehensive regulatory framework for crypto-asset markets. Fully in force since December 2024 (transition period's final deadline: July 2026). The scope of application to on-chain finance protocols is at the stage of watching detailed rulemaking, but the direction — rising demands for transparency and security — is a tailwind
- **PSA amendment (in force June 2026, Japan) / FIEA amendment:** the Payment Services Act amendment primarily targets crypto-asset exchange operators, but it may lead to stronger security requirements for on-chain finance access via exchanges. The Financial Instruments and Exchange Act (FIEA) amendment bill approved by the Cabinet in April 2026 sets a direction of reclassifying crypto-assets as "financial instruments" — a medium-term regime shift we are watching
- **EU AI Act:** in force since August 2024; **the key provisions — high-risk AI requirements, transparency obligations, and others — begin applying on August 2, 2026** (some phased in through 2027). In the Agentic Finance context, the explainability and auditability of AI-Agent behavior may come into regulatory scope in the future. Ninja's two-layer architecture (deterministic Layer 1 + LLM-augmented Layer 2) and its continuous evaluation of prompt-injection resistance (§ 5.2) are consistent with this regulatory direction

Taken together, we position both standardization (Clear Signing) and regulation (SEC, MiCA, the AI Act) as generating medium-term demand for "making things visible" — and, beyond it, for independent assessment and accountability.

Getting Started

You can try Ninja today.

- **NinjaScan / NinjaCheck (Telegram Bot):** send a contract address to @NinjaScanBot and a risk assessment comes back on the spot
- **Docs / API / MCP:** https://ninja.zksc.io/docs — MCP connects with no registration and anonymous access
- **PoC, partnership, and investor inquiries:** kaneshiro@zksc.io (ZKSC Inc. https://zksc.io)

References

1. DeFi Llama — Total Value Locked (TVL) Statistics. https://defillama.com/
2. Chainalysis — 2025 Crypto Crime Report. Hacking losses data ($3.4B).
3. Morpho Labs — Protocol metrics and growth data. https://morpho.org/
4. Hypernative — Company information and product documentation. https://hypernative.io/
5. Blockaid — Company information and product documentation. https://blockaid.io/
6. GoPlus Security — API documentation and company metrics. https://gopluslabs.io/
7. U.S. SEC Division of Trading and Markets — Staff Statement on DeFi Front-End Providers (April 13, 2026).
8. European Commission — Markets in Crypto-Assets Regulation (MiCA).
9. Financial Services Agency, Japan — Payment Services Act amendments.
10. European Commission — EU Artificial Intelligence Act.
11. Forta Network — Decentralized threat detection. https://forta.org/
12. Resolv Incident Report — March 2026. Compromised key exploit — unauthorized minting of 80M USR ($25M loss).
13. Web3 Security Market — $2.9B (2025) to $15.8B (2032 projected, CAGR 26.4%). Source: Electronics Media / Market Research analysis (March 2026).
14. Model Context Protocol (MCP) — Anthropic (2024). https://modelcontextprotocol.io/
15. Ethereum Foundation Blog — Clear Signing: Making Transaction Approvals Safer on Ethereum (May 12, 2026). https://blog.ethereum.org/
16. ERC-7730 — Structured Data Clear Signing Format (Draft; V2 updated April 2026). https://eips.ethereum.org/EIPS/eip-7730
17. ERC-8176 — Descriptor Attestations (referenced in EF Clear Signing announcement).
18. Lido — Kelp Incident Review (post-mortem, May 21, 2026).
19. x402 — An open protocol for internet-native payments (HTTP 402-based pay-per-call). https://www.x402.org/
20. Ninja / ZKSC — "Kelp recovered. But the graph that carried the cascade is still unmapped." (Medium, June 1, 2026).
21. Ninja / ZKSC — "Ethereum Clear Signing — Now Visible. Defensible Is Still Another Question." (Medium, June 9, 2026).

Ninja — Agentic Security Controlplane for On-chain Finance & Agentic Finance What You Read Is What You Sign. Humans today. Agents tomorrow. ShoGun — See Through On-chain Finance, Control Agentic Finance

Disclaimer: This whitepaper is provided for informational purposes only and does not constitute investment advice, a solicitation of financial products, or an offer of a token sale. The market data, competitive information, and regulatory developments described herein are based on publicly available information at the time of writing, and their accuracy is not guaranteed. Statements concerning regulatory and standardization developments do not constitute legal advice.