Flick: An AI Human-Assistant Agent for On-Chain Financial Security

The Flick Project $FLICK
flickonbase.com
t.me/flickonbase | x.com/flickonbase

Abstract. A purely peer-to-peer financial system has given individuals direct custody of their assets, but it has also removed the institutional safeguards that once sat between a user and a catastrophic mistake. Drainer contracts, malicious token approvals, address poisoning, and exploited protocols transfer billions of dollars of value irreversibly every year, and almost all of it is preventable by an analysis that completes in milliseconds. The tools to perform that analysis exist, but they are fragmented, built for professional auditors, and locked behind accounts, subscriptions, and API keys that ordinary users will not manage. We propose Flick, an AI human-assistant agent — a personal financial assistant that acts the way a trusted human advisor would, but autonomously and continuously — that inspects every transaction before it is signed, audits contracts and protocols on demand, and monitors a user's exposure continuously. The agent funds its own intelligence: each audit, simulation, and threat-intelligence request is a paid HTTP request settled per-call through the x402 protocol in stablecoins on an Ethereum Layer 2. This removes the account, key, and subscription friction that has kept security out of the default path, lets the agent's spending scale exactly with risk, and exposes security as a composable, machine-payable service that other autonomous agents can call before they sign. The result is a security layer that is opt-out rather than opt-in, and that participates in the same economy it protects.

1. Introduction

Self-custodial finance is built on the principle that a signature is final. This is its greatest strength and its most dangerous property. When a user signs a malicious approval or interacts with a drainer contract, there is no chargeback, no fraud department, and no settlement window in which the transfer can be reversed. The defenses that exist in traditional finance — a human reviewing a wire, a hold on a suspicious card, an institution absorbing fraud loss — have no native equivalent here.

What is needed is not another scanner. Scanners, simulators, and audit databases already exist. What is needed is a layer that sits in the default path of every transaction, applies that analysis automatically and before the signature, and does so without asking the user to create an account, hold an API key, or maintain a subscription. The reason such a layer does not exist by default today is largely economic: high-quality security intelligence costs money per query, and the prevailing way to pay for it — accounts and monthly plans — does not fit a model where most users need an answer a few times a day, unpredictably, and where the analysis must complete before a signature without a human in the loop.

We describe a system in which an autonomous agent performs this analysis and pays

for the intelligence it consumes on a per-request basis, at machine speed, using a payment primitive native to the network. The agent's economic footprint becomes a direct function of the user's risk exposure rather than a fixed subscription, and the same payment interface that the agent uses to buy intelligence allows other agents to buy a security verdict from it.

2. The Cost of an Irreversible Signature

Consider the lifecycle of a typical loss. A user is induced to sign a transaction — frequently an unlimited token approval, a transfer to a look-alike address, or a call into a contract whose true behavior differs from its presentation. The transaction is valid. It is included. The value is gone. The interval in which intervention is possible is the interval before the signature, and that interval is often measured in seconds because the user is acting inside a wallet prompt with limited information.

The analysis that would have prevented the loss is well understood: simulate the transaction against current state, decode and interpret the calldata, evaluate the counterparty against known-malicious labels and behavioral heuristics, and surface unlimited or unusual allowances. Each of these steps is individually tractable and, in aggregate, completes well within the window. The gap is not analytical capability. The gap is that this analysis is not in the path by default, because the economic and integration model for delivering it has not matched the way the risk actually arrives: occasionally, unpredictably, at machine speed, and without a human available to manage credentials.

3. Autonomous Security Agent

We define a Flick instance as a process that holds no custody of user funds and operates over three functions.

Guard. Before any transaction is signed, the agent simulates execution against current chain state, decodes the calldata, identifies unlimited or anomalous approvals, evaluates the counterparty, and returns a verdict — *allow*, *warn*, or *block* — accompanied by a plain-language explanation of the reasoning. The verdict is advisory; the user or a configured policy retains the decision.

Audit. On demand, the agent retrieves and synthesizes an assessment of a specified contract, token, or protocol. Because each audit is paid for individually at the moment of need (Section 5), the user incurs cost in proportion to actual use rather than carrying a standing subscription against infrequent need.

Watch. The agent continuously re-evaluates the user's standing exposure: open approvals, positions, and protocol dependencies. When a dependency is compromised or an approval becomes dangerous in light of new information, the agent escalates — alerting the user or, under explicit pre-authorized policy, acting — before value can move.

These functions are not novel in isolation. Their value here derives from being default, automatic, pre-signature, and — critically — self-funding, which is what the remainder of this paper addresses.

4. Local-First Operation

A security agent is a privileged observer of a user's financial life. It must not become a new custodial dependency or a centralized aggregation point for the exact data it protects. Flick is therefore local-first: the simulation engine, the set of monitored addresses, audit history, and risk context reside on the user's machine, encrypted at rest. The agent

never takes custody of funds. Remote services are consulted only for discrete units of intelligence — a single simulation, a single audit, a single threat lookup — and only the minimal query required for that unit leaves the device. The user's aggregate financial graph is never assembled by a third party as a precondition of being protected.

5. Self-Funding via Per-Request Payment

The central mechanism of the system is that the agent pays for the intelligence it consumes, autonomously, one request at a time.

External intelligence — contract audits, transaction simulation, threat-graph lookups, model inference — is exposed by providers as HTTP endpoints that require payment per call. An unpaid request returns the HTTP status `402 Payment Required` together with the price and the accepted settlement instrument. The agent, holding a bounded balance in a stablecoin on an Ethereum Layer 2, constructs and signs the required payment authorization, attaches it to the request, and receives the response in the same exchange. Settlement is on-chain and final; no account is created, no API key is held, and no relationship persists beyond the single transaction.

Let a protective action require a set of intelligence calls $\{c_1, c_2, \ldots, c_n\}$ with prices $\{p_1, p_2, \ldots, p_n\}$. The cost of that action to the user is

$$C \;=\; \sum_{i=1}^{n} p_i,$$

and the agent's total economic footprint over a period is simply the sum of C over the protective actions actually taken. There is no fixed term. A user who transacts rarely funds little; a user under active threat funds more, precisely when the protection has the most value. Expenditure is bounded operationally by a user-configured budget B, such that the agent will not initiate a call when cumulative spend would exceed B without explicit escalation. This converts security from a fixed cost paid in anticipation of risk into a variable cost incurred in proportion to it.

6. Security as a Machine-Payable Service

The payment interface the agent uses to *purchase* intelligence is symmetric: the agent can also *sell* a verdict through the same mechanism. An autonomous agent — a trading agent, a treasury manager, another assistant — can submit a candidate transaction to a Flick endpoint, receive a `402` challenge, settle it, and obtain a security verdict before it signs.

This makes pre-signature security a composable primitive rather than a human-operated product. In an environment where an increasing share of transactions are initiated by software acting under delegated authority, the ability for one agent to obtain a priced, machine-verifiable safety assessment from another, with no account provisioning and no shared secret, is a structural requirement for that environment to be safe. Flick is designed to be both a consumer and a provider in that market.

7. Incentives

The model aligns three parties without trusted intermediation. Users obtain default, pre-signature protection and pay only in proportion to the risk they actually incur. Intelligence providers — auditors, simulation services, threat-data maintainers — receive

direct, immediate, per-query revenue for work that is otherwise difficult to monetize at the granularity at which it is consumed, which incentivizes the supply and continued maintenance of accurate data. Operators of Flick endpoints earn per-verdict revenue from agent traffic, incentivizing availability and accuracy, since a provider that returns poor verdicts loses callers in a market with no lock-in. Because settlement is per-call and final, no party need extend credit or custody to another, and the incentive to maintain quality is continuous rather than contractual.

8. Token and Distribution

The protocol token, `$FLICK`, exists to coordinate the security market described above rather than to capture rent from it. Its core utility is direct and singular: `$FLICK` is the settlement and access instrument of the Flick dApp. It is the unit in which the agent funds its own intelligence under the x402 mechanism of Section 5 — each Guard simulation, on-demand Audit, and Watch escalation is a per-call payment denominated through `$FLICK` — and it is the unit in which external agents pay to obtain a security verdict from a Flick endpoint under Section 6. Demand for the token is therefore a function of protective activity actually performed, not of speculation layered on top of it: every unit of intelligence the network consumes or sells flows through it.

Distribution is structured as a fair launch with no team, investor, or pre-allocated treasury tranche:

- **95% — Liquidity.** The overwhelming majority of supply is placed directly into the on-chain liquidity pool at launch, ensuring an open, deep, and permissionless market in which the token's per-call utility can be exercised by any user or agent without gatekeeping.
- **5% — Marketing.** A minimal reserve is retained solely to fund distribution, integrations, documentation, and awareness, so that the security layer reaches the users and agents whose risk it is designed to reduce.

There is no founder allocation, no vesting cliff held over the market, and no discretionary treasury. The design intent is that the token's value, if any, derives strictly from its function as the metering instrument of a security service that is consumed per call — consistent with the per-action cost model $C = \sum_i p_i$ of Section 5 — and not from a privileged distribution to insiders.

9. Core Utility of the dApp

To state the product precisely: the Flick dApp is a non-custodial, local-first AI human-assistant agent that a user installs and connects to their wallets, and that acts as a personal financial-security assistant on their behalf. Its core utility, in one sentence, is to make every on-chain action safe by default by inspecting it before signature, auditing what it touches, and watching what it has touched — and to pay for the intelligence required to do so autonomously, per call, through the token. Concretely, the dApp delivers three user-facing capabilities, each metered through `$FLICK`: a pre-signature transaction Guard that returns an allow/warn/block verdict with a plain-language reason; an on-demand Audit that assesses any contract, token, or protocol at the moment of need; and a continuous Watch that monitors standing approvals and protocol exposure and escalates before value can move. A fourth capability is structural rather than user-facing: the same paid endpoint that lets the agent buy intelligence lets other autonomous agents buy a verdict from it,

making the dApp both a consumer and a provider in the machine-payable security market. This is the entirety of the product; everything else in this paper is the economic and privacy argument for why it is built this way.

10. Privacy

Per-call settlement and local-first operation jointly narrow the privacy surface. No provider holds a durable account that accumulates a history of a user's queries; each request is discrete and minimal, and the agent can vary the settlement address across requests so that a single provider does not reconstruct the user's full activity from payment metadata. The user's complete financial graph — the asset the agent exists to protect — is never the unit of exchange. Only the specific artifact under evaluation at a given moment is ever transmitted, and only to obtain the single answer required.

11. Calculations

We consider whether a rational user adopts default, self-funding protection. Let an unprotected transaction carry probability q of a loss event with expected severity L. The expected loss without protection is qL. With the agent in path, let d be the probability that an actual threat is correctly classified and stopped, and let C be the per-action cost from Section 5. The expected loss with protection is approximately

$$(1 - d)\, qL \;+\; C.$$

Protection is rational whenever

$$C \;<\; d\, qL.$$

Because C is a small per-call sum and L for the relevant loss classes is frequently the entire approved or transferred balance, the inequality holds by a wide margin for any non-negligible d and q. The decisive design consequence is not the size of the margin but its structure: since C is incurred per action rather than as a fixed subscription, the user is never in the position of paying for protection during the long intervals in which no risk is present, which is the precise condition under which users historically decline to adopt security and revert to the unprotected default.

12. System Architecture

A Flick instance is composed of a local core and a set of priced remote intelligence sources reached only through the payment interface of Section 5. The local core holds the user's policy, the set of monitored addresses, the budget accumulator, and the encrypted history; it never delegates custody and never exports the aggregate financial graph. Figure 1 shows the path of a single transaction through the system.

The Guard component is in-path: it receives the candidate transaction before the signature is released, fans out the bounded set of intelligence calls it needs, and blocks the signature until a verdict is produced or the policy explicitly permits proceeding under uncertainty. The Watch component is out-of-path: it runs on an interval, re-derives the user's standing exposure, and raises a Guard-equivalent evaluation when a previously safe state becomes unsafe. The Audit component is invoked explicitly by the user or by another agent. All three share the same budget accumulator, so the user's total expenditure is a single governed quantity rather than three independent meters.

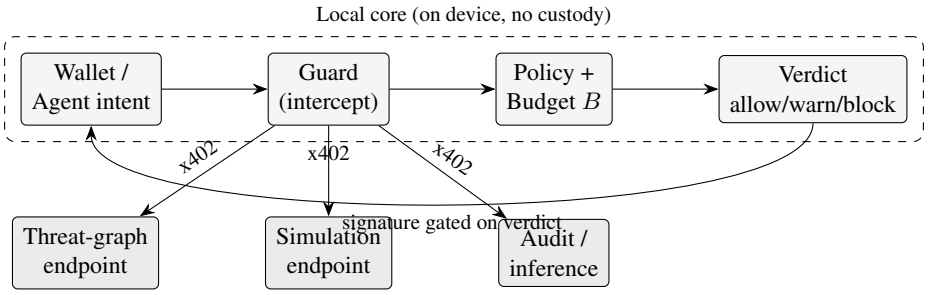

Figure 1: Transaction path. Every external call is metered per request via x402; the signature is gated on the returned verdict and the budget accumulator.

13. The Payment Flow

The novel mechanism of the system is the per-request settlement loop, so we specify it concretely. Let the agent hold a balance b in a stablecoin on an Ethereum Layer 2 and a user-configured budget B with cumulative spend s.

1. The agent issues an unpaid request to an intelligence endpoint for a discrete unit of work (one simulation, one audit, one threat lookup).
2. The endpoint responds with `402 Payment Required`, a price p, an accepted asset, and a destination.
3. The agent checks the gate: proceed only if $s + p \leq B$ and $p \leq b$. If the gate fails, the agent does not pay; it returns an explicit *insufficient-budget* state to the policy rather than silently degrading.
4. The agent constructs and signs a payment authorization for exactly p, attaches it to a re-issued request, and the endpoint returns the result in the same exchange. Settlement is on-chain and final.
5. The accumulator updates $s \leftarrow s + p$. No account persists; the next request to the same endpoint repeats the loop from step 1.

Three properties follow directly. First, spend is *bounded by construction*: the gate at step 3 makes $s \leq B$ an invariant the agent cannot violate, which is the property that makes it safe to grant an autonomous process a wallet at all. Second, there is *no trust extension*: because settlement precedes delivery and is final, neither party carries counterparty credit risk, which is what allows the network of providers to be open and permissionless rather than contractually onboarded. Third, the loop is *symmetric*: a Flick endpoint serving verdicts to other agents runs the same protocol with the roles reversed, which is what makes a verdict a sellable unit (Section 6) without any additional machinery.

14. Threat Model and Limits

A security document that does not state what it cannot do should not be trusted. Flick is designed to reduce loss from transaction-time deception: malicious or unlimited approvals, drainer and honeypot contracts, address poisoning and look-alike destinations, calldata whose effect differs from its presentation, and exposure to a dependency that becomes compromised while the user holds a position. Against these, an in-path, pre-signature, simulated evaluation is strong because the adversary's success depends on the user not performing exactly the analysis the agent performs automatically.

The system is explicitly *not* a guarantee. A novel exploit absent from any consulted intelligence source may pass; detection probability d in Section 11 is below one by assumption, not by omission. The agent's verdict is only as good as the intelligence it can purchase, which is why the economic design deliberately incentivizes a competitive supply of providers (Section 7) rather than relying on a single source. Compromise of the user's device is out of scope: a local-first design protects the user's data from third parties but assumes the endpoint where the user makes decisions is theirs. Finally, the agent gates a signature; it does not seize one. Under default policy a user may override a *warn*, by design, because a security tool that cannot be overridden becomes a custody arrangement, which is precisely the property self-custodial finance exists to avoid. These limits are stated because a defensible security claim is a narrow one.

15. Adoption and Network Effects

The system's growth argument is structural, not promotional, and rests on a two-sided market that compounds.

On the demand side, every additional autonomous agent that signs transactions is a potential caller of a security verdict (Section 6). The fraction of on-chain activity initiated by software acting under delegated authority is increasing; each such agent that signs without a pre-signature check is an uninsured liability to whoever delegated to it. As the cost of a verdict is a small per-call sum and the loss it averts is frequently a full balance (Section 11), the rational default for any non-trivial agent is to call before it signs. Demand for verdicts therefore scales with the population of transacting agents, not with marketing.

On the supply side, per-call settlement makes serving verdicts and serving intelligence directly revenue-generating at the granularity at which they are consumed, with no onboarding and no lock-in. This is precisely the condition under which independent providers enter: the marginal provider is paid immediately and competes on verdict quality because callers, having no contract, move freely. More providers raise aggregate detection probability d, which raises the value of a verdict, which raises demand, which attracts further supply. The loop is self-reinforcing and, critically, the token sits in the middle of every turn of it (Section 8): each unit of intelligence consumed or sold is denominated through `$FLICK`, so utility demand for the token is a monotone function of the security activity the network performs. This is the substantive case for the project's trajectory — a compounding two-sided market with the token as its settlement layer — stated without recourse to claims the document cannot support.

16. Protocol Sustainability

Because distribution is a fair launch with no treasury (Section 8), the protocol must be self-sustaining from function rather than from a reserve. It is. The per-verdict revenue that accrues to Flick endpoint operators (Section 7) funds the operation of those endpoints directly; sustainability is therefore a property of usage, not of a runway that can be exhausted. A minimal, governance-set protocol fee on agent-to-agent verdict settlement may be directed to a public, on-chain pool that funds shared intelligence and integration work, so that the commons the network depends on is maintained without reintroducing a privileged treasury or a discretionary insider allocation. The design principle is consistent throughout: the system pays for itself out of the activity it performs, at the granularity at which it performs it.

17. Roadmap

Development proceeds in stages, each of which is independently useful so that the system delivers protection before it is feature-complete.

- **Stage I — Guard.** In-path pre-signature evaluation with x402-metered simulation and threat lookup on an Ethereum Layer 2; local-first core; bounded budget accumulator.
- **Stage II — Audit and Watch.** On-demand auditing of arbitrary contracts and continuous monitoring of standing exposure, sharing the Stage I budget and policy.
- **Stage III — Verdict-as-a-service.** The symmetric endpoint: external agents obtain priced verdicts before signing, making the dApp a provider as well as a consumer in the security market.
- **Stage IV — Open provider market.** A permissionless set of competing intelligence and verdict providers, governed quality signaling, and the sustainability pool of Section 16.

18. Conclusion

We have described a security layer for self-custodial finance that is default rather than opt-in, pre-signature rather than post-mortem, and self-funding rather than subscription-bound. The agent inspects transactions before they are signed, audits and monitors on demand, and operates local-first without taking custody. Its defining property is economic: by paying for each unit of intelligence per request, at machine speed, using a settlement primitive native to the network, the agent's cost tracks the user's risk exactly, the account and key friction that kept security off the default path is removed, and the same interface turns a verdict into a service that other autonomous agents can purchase before they sign. As an increasing fraction of on-chain activity is initiated by software, a security primitive that is itself machine-payable and machine-callable is not an enhancement but a precondition.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.

[2] Coinbase, "x402: An open standard for internet-native payments using the HTTP 402 status code," technical specification, 2025.

[3] J. Benet, "IPFS — Content Addressed, Versioned, P2P File System," 2014.

[4] Ethereum Foundation, "Layer 2 scaling and rollup execution," technical documentation.

[5] OpenHuman, "A local-first personal AI agent with on-device memory," project documentation, 2026.

[6] Threat-intelligence and transaction-simulation literature for EVM environments (drainer detection, approval risk, address poisoning), surveyed.