// WHITEPAPER v1.0 · APRIL 2026

SCRUTOR

Web3 AI Agent Infrastructure on Ethereum

Autonomous AI agents, natively secured on-chain.
Wallet-bound identity. Skill-Guard threat scanning.
Free, open, and production-grade — built on Ethereum.

AGENTS	P95 LATENCY	UPTIME SLA	AGENT CALLS
25+	< 50ms	99.99%	2.4M+
deployed on-chain	ETH · L2 network	production-grade	past 30 days

VERIFIED SOURCES Alchemy · DefiLlama · OpenRouter · Coinbase · Ethereum · Arbitrum · Optimism · Etherscan · Basescan · 1inch

scrutor.org · Ethereum Network · MIT Licensed CLI & SDK · On-chain Verified Contracts

Version 1.0 · April 2026

// NAVIGATION

Table of Contents

// SECTION 01

Executive Summary

Scrutor is a Web3-native AI agent infrastructure designed to make autonomous, multi-step AI agents free, open, and production-ready for anyone with an EVM-compatible wallet. By replacing traditional API keys with cryptographic wallet identities on Ethereum, Scrutor eliminates server-side credential management while delivering deterministic, auditable access control on-chain.

At the core of Scrutor sit three interlocking innovations: (1) wallet-bound agent runtimes that derive API credentials directly from the user's Ethereum signing key, (2) the skill.md specification — a portable, markdown-native format for defining agent prompts and tool integrations — and (3) Skill-Guard, a six-detector security pipeline that scans every skill for credential leaks, prompt injection, malicious commands, data exfiltration, permission abuse, and phishing URLs in under 50 milliseconds.

The network is free at the base layer. Users pay nothing to call agents, receive responses via server-sent events (SSE), and compose any number of skills without vendor lock-in. The $SCTR token governs the protocol: skill authors stake to publish, model operators earn from usage emissions, and the community votes on routing, moderation, and upgrades.

SKILL SCANS	THREATS BLOCKED	COMMUNITY FLAGGED	AVG SCAN LATENCY
10,000+	500+	36%	< 50ms
cumulative	by Skill-Guard	flagged on first submit	six detectors parallel

// SECTION 02

Introduction & Problem Statement

2.1 The AI Agent Landscape

The proliferation of large language models has given rise to a new class of software: autonomous AI agents. Unlike simple chatbots, agents reason over multi-step plans, invoke external tools, manage state across turns, and execute consequential actions — transferring assets, querying on-chain state, and triggering smart contract calls on Ethereum. The market for agent infrastructure is nascent but accelerating. Most existing platforms, however, are built on Web2 primitives: API keys emailed to inboxes, billing dashboards linked to credit cards, and identity anchored to centralised OAuth providers — fundamentally misaligned with the self-sovereign ethos of Web3 and Ethereum.

2.2 The Web3 Identity Problem

When a user signs an Ethereum transaction, that signature is cryptographically unforgeable — rooted in secp256k1 elliptic curve cryptography on a device they control. Yet when the same user calls an AI agent API, they receive a UUID string stored in a third-party database. This disconnect creates several failure modes:

- Credential exposure: API keys are plaintext secrets. A leaked environment variable drains budgets and access.
- Centralised revocation: A provider database breach simultaneously invalidates all user keys.
- No on-chain auditability: There is no way to verify, on Ethereum, which agent called which skill on whose behalf.
- Subscription lock-in: Usage is metered against a stored payment method, creating involuntary dependency.

Scrutor resolves this gap by deriving agent credentials deterministically from the user's Ethereum wallet signing key. The result (format: sctr_live_*) is always recoverable, never stored server-side, and verifiable on-chain.

2.3 The Security Gap in Agent Skills

As skill registries grow, so does the attack surface. Malicious skill authors can embed prompt injection payloads, exfiltrate environment variables through tool definitions, or ship reverse-shell commands inside skill bodies. Unlike traditional package registries — where static analysis tooling is mature — the agent skill ecosystem has no equivalent of a standardised vulnerability scanner for prompt-level threats. Scrutor's Skill-Guard addresses this gap directly, running six security detectors on every skill submission before it reaches the public registry.

// SECTION 03

Architecture Overview

3.1 Core Design Philosophy

Scrutor is built around four architectural tenets enforced at the protocol level:

- **Wallet-first identity**
  Authentication derives from cryptographic proof on Ethereum, not shared secrets. No email, no password, no OAuth.
- **Zero-cost base layer**
  The public agent network is free by design. Premium features are additive, not gating.
- **Skill portability**
  The skill.md specification is an open standard — plain Markdown, no proprietary DSL, no vendor-specific container.
- **Security-first publishing**
  Every skill is scanned before publication. The registry is a curated trust layer, not a raw upload endpoint.

3.2 System Layers

Layer 5 — Application	Agent Chat UI, CLI, MCP Preview, OpenAPI clients
Layer 4 — SDK / API	TypeScript SDK, REST /v1/* endpoints, SSE streaming gateway
Layer 3 — Skill Runtime	skill.md parser, tool dispatcher, multi-model router (25+ models)
Layer 2 — Security	Skill-Guard six-detector pipeline, CWE-mapped threat reporting
Layer 1 — Identity & Chain	Ethereum wallet auth, deterministic key derivation, on-chain manifest

3.3 Network Topology

Scrutor operates natively on Ethereum, with additional L2 support for Arbitrum and Optimism. Agent calls are routed through an edge layer optimised for low-latency SSE delivery. The routing layer aggregates across 25+ model backends, applying skill-defined model preferences and automatic failover. Smart contracts are deployed and verified on Etherscan, serving as the authoritative source for the on-chain skill registry manifest.

// SECTION 04

Autonomous Agents

4.1 Agent Runtime

The Scrutor agent runtime is a stateful execution environment that wraps a multi-turn reasoning loop around any skill. Each agent instance is wallet-bound — tied to a specific Ethereum signing address — ensuring that tool calls, resource access, and result attribution are traceable on-chain.

Runtime Environment	Edge + SSE
Auth Model	Ethereum wallet (EIP-191 signed challenge)
P95 Latency	< 50ms
Streaming Protocol	Server-Sent Events (SSE) + JSON
SDK Languages	TypeScript, CLI (bash-compatible)
Network	Ethereum · Arbitrum · Optimism
Agent Status	LIVE

4.2 Agent Types & Capabilities

Scrutor currently hosts 25 deployed agents across multiple categories:

- **DeFi Agents** — On-chain price queries, swap routing, and liquidity analysis via Alchemy, 1inch, and DefiLlama.
- **Wallet Intelligence** — Transaction history analysis, portfolio valuation, and risk scoring across Ethereum and L2 chains.
- **Market Data Agents** — Real-time price feeds, volume analytics, and alert-triggered actions via Coinbase and Etherscan.
- **Governance Agents** — DAO proposal summarisation, vote delegation, and quorum tracking on Ethereum.
- **Security Audit Agents** — Smart contract analysis, bytecode inspection, and vulnerability pattern matching.
- **Cross-Chain Agents** — Bridging intelligence across Ethereum mainnet, Arbitrum, and Optimism.

4.3 Multi-Step Execution

Scrutor agents implement a plan-execute-observe loop. Given a natural language goal, the agent produces a step-by-step execution plan, invokes required tools in sequence (or in parallel where dependencies allow), observes intermediate results, and revises its plan before producing a final streamed response.

1. RECEIVE →	2. PLAN →	3. EXECUTE →	4. OBSERVE →	5. STREAM
User prompt arrives	Task decomposed into tool calls	Skills invoked in optimal order	Results evaluated; plan revised	Final answer streamed token-by-token

// SECTION 05

Wallet-Based Authentication

5.1 Deterministic Key Derivation

When a user connects their Ethereum wallet, Scrutor presents a deterministic challenge message for signing via EIP-191. The resulting signature — combined with the user's Ethereum address and a protocol-specific salt — is passed through a one-way derivation function to produce the API credential. The format is sctr_live_* and is always recoverable from the same wallet without any server-side state.

```
// Key derivation (simplified)
const key = keccak256(ethAddress + protocolSalt + chainId)
return 'sctr_live_' + key.slice(0, 32)
```

5.2 Zero-Server Identity

Scrutor stores no credentials. The derivation function is public and deterministic — the platform itself cannot forge a credential for an Ethereum address it does not control. This property is formally equivalent to the security of secp256k1, the same curve securing all Ethereum transactions. Revocation requires only a new wallet signature — no support ticket, no database update.

5.3 Supported Networks

Ethereum Mainnet	**PRIMARY**	Agent calls, skill publishing, $SCTR contract, governance voting.
Arbitrum	**SUPPORTED**	Cross-chain agent data and bridging intelligence.
Optimism	**SUPPORTED**	Cross-chain agent data and bridging intelligence.
More L2s	**ROADMAP**	Additional Ethereum L2 networks planned post v2.

// SECTION 06

The Skill System

6.1 The skill.md Specification

A skill is a single Markdown file that fully specifies an agent's behaviour. No proprietary DSL, no binary format, no framework dependency. Any text editor, version control system, or CI pipeline can read, diff, and publish a skill. A skill.md contains four sections:

`--- Frontmatter ---`

YAML block defining name, version, model preference, network scope, and required permissions.

`## System Prompt`

Persistent agent instructions. Supports Jinja2-style variable interpolation for dynamic context injection.

`## Tools`

Tool definitions — name, description, input schema, endpoint. Compatible with the OpenAI function-calling schema.

`## Examples`

Optional few-shot examples to guide model output format. Versioned alongside the skill for reproducibility.

6.2 On-Chain Registry

Every skill published through the Scrutor CLI is hashed, signed by the author's Ethereum wallet, and committed to the on-chain registry contract (verified on Etherscan). The manifest stores: skill hash, author address, Skill-Guard scan result hash, timestamp, and staked $SCTR amount. Any modification to a published skill produces a different hash and invalidates the on-chain record — the registry is tamper-evident by design.

6.3 Publishing Workflow

01

Author skill.md

Write a prompt + tool spec in plain Markdown. Any editor works.

02

Run scrutor scan

Skill-Guard executes all six detectors in parallel. CWE-mapped results with line-level annotations are returned in < 50ms.

03

Sign + publish

CLI packages the skill, prompts for Ethereum wallet signature, and broadcasts the transaction. Live in < 15 seconds.

// SECTION 07

Skill-Guard Security Framework

7.1 Pipeline Overview

Skill-Guard is Scrutor's write-time security pipeline. Every skill targeting the Ethereum on-chain registry passes through six parallel detectors before the author receives a publish token. The pipeline completes in under 50ms at P95 — fast enough for CI/CD integration without meaningful developer overhead.

CRITICAL severity findings block publish entirely. HIGH and MEDIUM findings produce a warning report with line-level annotations — the author decides whether to ship. Detector rulesets are open, versioned, and subscribable. Teams can pin a ruleset hash for reproducible scans.

SKILLS SCANNED	THREATS BLOCKED	SCAN LATENCY	FLAGS RATE
10K+	500+	< 50ms	36%
cumulative	CRITICAL severity	P95, six detectors	community first-submit

7.2 The Six Detectors

DETECTOR 01

CRITICAL

Credential Leak Detector

Matches against 160+ secret patterns covering API keys, Ethereum private keys, JWT tokens, BIP-39 mnemonics, AWS credentials, and service-account JSON blobs. Operates on skill bodies and embedded tool endpoint headers. Any match at CRITICAL severity blocks publish.

Coverage

160+ patterns

CWE

CWE-200

DETECTOR 02

HIGH

Prompt Injection Detector

Applies 75+ heuristics to identify jailbreak attempts, role-swap attacks, goal-hijacking payloads, and indirect injection through tool outputs. Covers obfuscation variants including Unicode homoglyphs, base64-encoded instructions, and whitespace manipulation.

Coverage

75+ heuristics

CWE

CWE-77

DETECTOR 03

CRITICAL

Malicious Command Detector

Scans skill bodies for 112+ patterns covering RCE, reverse shells, post-exploit cleanup sequences, and dangerous shell idioms (eval, exec, curl-pipe-bash). Targets both explicit command strings and obfuscated equivalents.

Coverage

112+ patterns

CWE

CWE-78

DETECTOR 04

HIGH

Data Exfiltration Detector

Detects path traversal sequences, environment variable harvesting (process.env, os.environ), DNS tunneling in tool endpoint URLs, and outbound webhook drops that could leak session context or Ethereum wallet state to attacker-controlled infrastructure.

Coverage

Path + net heuristics

CWE

CWE-538

DETECTOR 05

MEDIUM

Permission Abuse Detector

Diffs declared permissions in the skill frontmatter against resources the skill body actually accesses. Flags privilege creep — skills claiming read-only access that embed write operations, or narrow network scope that call unrestricted endpoints.

Coverage

Scope diffing

CWE

CWE-269

DETECTOR 06

HIGH

URL & Domain Risk Detector

Cross-checks every URL in the skill against live phishing feeds, typosquat detection, and a known-malicious infrastructure blocklist. Flags domains impersonating legitimate services or appearing on community-maintained threat intelligence lists.

Coverage

Live threat feed

CWE

CWE-601

7.3 CWE Mapping

All Skill-Guard findings are reported against the Common Weakness Enumeration (CWE) taxonomy maintained by MITRE — giving security teams a standardised vocabulary compatible with existing vulnerability management workflows. The mapping is updated with each detector ruleset release.

7.4 Scan Performance

All six detectors run in parallel within a single scan job. The P95 latency target of 50ms is enforced by a per-job timeout: if any detector exceeds its budget, it returns a partial result flagged INCOMPLETE rather than blocking the pipeline. In practice, fewer than 0.1% of scans trigger this fallback.

// SECTION 08

Streaming Infrastructure

8.1 SSE Protocol

Agent responses are delivered via Server-Sent Events (SSE) — a lightweight, HTTP/1.1-compatible streaming protocol with lower connection overhead than WebSockets, automatic reconnection, and native browser support. Each event carries a JSON payload: token delta, event type, and sequence number for client-side reordering.

```
data: {"type":"token.delta","seq":42,"delta":"Swap route found: "}
data: {"type":"token.delta","seq":43,"delta":"USDC → ETH via 1inch"}
data: {"type":"complete","seq":44,"usage":{"prompt":182,"completion":47}}
```

8.2 SDK & CLI

Scrutor ships a TypeScript SDK and a bash-compatible CLI, both MIT-licensed. The SDK exposes an async iterator interface over the SSE stream with automatic reconnection, event parsing, and error propagation. The CLI is designed for scripting and CI integration.

```
import { ScrutorClient } from '@scrutor/sdk'

const client = new ScrutorClient({ wallet })
const stream = client.agents.stream({ skill: 'defi-snipe', network: 'ethereum' })

for await (const token of stream) {
process.stdout.write(token.delta)
}
```

8.3 Performance Benchmarks

Metric	Target	Measured P95
Time to first token	< 800ms	**620ms**
Inter-token latency	< 30ms	**18ms**
SSE connection setup	< 150ms	**110ms**
Agent skill load	< 50ms	**12ms**
End-to-end (short task)	< 3s	**1.8s**
End-to-end (multi-step)	< 12s	**8.4s**

// SECTION 09

Data Sources & Integrations

Scrutor agents operate on verified, production-grade data sources. All integrations are configured at the skill level — no global API key rotation, no shared credential pool. Each source is vetted for reliability, rate-limit transparency, and on-chain verifiability where applicable.

Alchemy	RPC & NFT API	Primary node provider for Ethereum and L2s. On-chain state queries, transaction simulation, block data.
DefiLlama	DeFi Analytics	TVL tracking, protocol revenue, and yield data across 200+ protocols. Used by DeFi intelligence agents.
OpenRouter	Model Aggregation	Routes inference across 25+ LLM backends. Provides automatic failover and cost optimisation.
Coinbase	Market Data	Real-time and historical price data for 500+ assets. Used by market data and trading agents.
Ethereum	L1 Chain	Primary execution and settlement layer. $SCTR contract, skill registry, governance.
Arbitrum	L2 Chain	Cross-chain agent data and bridging intelligence on Ethereum's leading L2.
Optimism	L2 Chain	Cross-chain agent data and bridging intelligence on the OP Stack.
Etherscan	Block Explorer	Contract verification, ABI resolution, and transaction receipt queries on Ethereum.
Basescan	Block Explorer	On-chain skill registry manifest verification and additional contract auditing.
1inch	DEX Aggregator	Optimal swap routing across decentralised exchanges for Ethereum DeFi agents.

// SECTION 10

Developer Experience

10.1 Three-Step Quickstart

Scrutor is designed for a zero-to-first-call time of under sixty seconds.

STEP 01

Connect Wallet

Navigate to scrutor.org and connect any EVM-compatible wallet (MetaMask, Coinbase Wallet, WalletConnect). Sign the deterministic Ethereum challenge. Your API credential (sctr_live_*) is instantly derived — no email, no password, nothing stored server-side.

```
$ scrutor auth
✓ Wallet connected
✓ API key: sctr_live_••••1a2b
```

STEP 02

Pick a Skill

Browse the on-chain skill registry at scrutor.org/agents or via the CLI. Each skill displays its Ethereum author address, Skill-Guard scan score, and usage metrics. Select a skill and it is instantiated as a live agent endpoint.

```
POST /v1/agents
→ skill: defi-snipe
→ network: ethereum
✓ ready (12ms)
```

STEP 03

Stream Responses

Call GET /v1/stream with your derived API key. Responses arrive token-by-token over SSE. Use the TypeScript SDK for production integrations or the CLI for scripting and testing.

```
GET /v1/stream
← token.delta …
← token.delta …
✓ complete
```

10.2 API Reference

The Scrutor REST API follows OpenAPI 3.1. Full specification at scrutor.org/cli.

POST	POST /v1/agents	Instantiate an agent from a skill ID.
GET	GET /v1/stream	Open an SSE stream for an active agent session.
GET	GET /v1/skills	List the on-chain skill registry with filtering.
POST	POST /v1/skills	Publish a new skill (requires Ethereum wallet signature + scan pass).

GET	GET /v1/skills/{id}/scan	Retrieve Skill-Guard results for a skill.
GET	GET /v1/status	Network health and live telemetry.

10.3 Self-Hosting

Authors can write a skill.md pointing to their own model endpoint, publish to the Ethereum on-chain registry, and receive routing from the Scrutor network without exposing their infrastructure. The enterprise tier ships with a self-hosted runtime container, private skill namespaces, and dedicated SSE gateway capacity.

// SECTION 11

Open Source & Licensing

The following Scrutor components are publicly available and community-maintainable:

CLI	MIT	Full source. Install globally via npm or use without installation via npx.
TypeScript SDK	MIT	Published as @scrutor/sdk. Full type definitions included.
skill.md Specification	MIT	Open standard for agent skill definitions. Fork and extend freely.
Routing Layer	Source-available	Available for audit. Not forkable for commercial reuse without agreement.
Smart Contracts	Verified on Etherscan	Contract addresses published. ABI downloadable from Etherscan.
Detector Rulesets	Open + Versioned	CWE-mapped rule definitions. Subscribe to changelog or pin a hash.

// SECTION 12

Roadmap

The Scrutor roadmap is organised into three phases, each building on the previous while preserving the zero-cost, wallet-first, Ethereum-native foundation.

PHASE 1 — LIVE — Foundation

- → 25+ production agents on Ethereum
- → Skill-Guard six-detector security pipeline
- → Ethereum wallet-bound authentication and deterministic key derivation
- → SSE streaming gateway with TypeScript SDK and CLI
- → On-chain skill registry verified on Etherscan
- → OpenRouter model aggregation (25+ backends)

PHASE 2 — IN PROGRESS — Expansion

- → Additional L2 network support (Polygon, zkSync, Linea)
- → MCP (Model Context Protocol) full integration and GA
- → Skill composability — chain skills into agent pipelines
- → Agent-to-agent communication (A2A) protocol
- → Private skill namespaces for enterprise teams
- → Skill marketplace with Ethereum-native attribution tracking

PHASE 3 — PLANNED — Protocol Maturity

- → $SCTR fair launch on Ethereum with 0/0 tax
- → DAO governance for model routing and skill moderation
- → Weekly $SCTR emissions tied to skill usage (contributor rewards)
- → Skill staking and on-chain reputation layer
- → Fully self-hosted enterprise runtime GA
- → Cross-chain agent execution via Ethereum intent-based bridging

// SECTION 13

Conclusion

Scrutor represents a fundamental rethinking of how AI agent infrastructure should be built for the Ethereum era. By anchoring identity in cryptographic wallet primitives, publishing agent skills on-chain, and enforcing security at write-time rather than runtime, Scrutor creates an agent ecosystem that is open by default, secure by design, and owned by its participants.

The network is live. Twenty-five agents are deployed on Ethereum. Skill-Guard has scanned over 10,000 skills and blocked over 500 threats. The P95 latency is under 50ms. The uptime SLA is 99.99%. And the entire base layer is free — no credit cards, no subscriptions, no metering.

For developers, Scrutor offers a path from Ethereum wallet connect to production agent in under sixty seconds. For security teams, it offers the first standardised, CWE-mapped threat scanner for AI agent skills. For the broader Web3 community, it offers a governed, community-owned alternative to centralised AI provider lock-in — built natively on Ethereum.

The next phase — expanded L2 support, full MCP integration, skill composability, and the $SCTR fair launch on Ethereum — will deepen the protocol's capabilities while preserving the principles that define it: zero cost, wallet-first, open, and on-chain.

GET STARTED

scrutor.org

Open Agent Chat · OpenAPI Spec · MCP Preview

DEVELOPERS

scrutor.org/cli

CLI · SDK · API Reference · Status

ETHEREUM

Ethereum Network

Etherscan Verified · MIT Licensed · On-chain