

Circle Agent Stack vs Verigate

Overlap Analysis, Pivot Strategy & Implementation

Prepared: August 02, 2026

Based on: *The Circle Agent Stack and Agent Wallets: Architecture, Infrastructure, and the Agentic Economy* (detailed report, mid-2026)

This document analyzes the overlap between Circle's Agent Stack and Verigate, explains the strategic pivot from policy enforcement to cryptographic proof, and documents the new features implemented to differentiate Verigate as the missing compliance and forensics layer for Circle's ecosystem.

1. The Problem: Our Old Pitch Was Redundant

Old pitch: "Verigate stops AI agents from spending money they shouldn't."

A judge who reads Circle's Agent Stack report would immediately see that Circle's Action Gate + MPC co-signer already does this at the infrastructure layer. The co-signer independently refuses to sign transactions that violate spending caps, allowlists, or sanctions. Circle also has Input Guardrails (pre-LLM), Output Guardrails (post-LLM), and Firecracker microVM isolation per agent session.

Every feature we were leading with -- spending caps, allowlists, rate limits, prompt injection defense, agent isolation -- is now something Circle does natively. Positioning these as our differentiators would make us look like we don't understand Circle's stack.

Feature	Circle's Version	Verigate's Old Version	Overlap
Spending caps	Action Gate + MPC co-signer enforces at cryptographic layer	Python PolicyEngine checks amount before CLI call	HIGH
Allowlists	Action Gate + MPC co-signer checks destination addresses	PolicyEngine resource_scope rule	HIGH
Rate limits	Wallet-layer transaction volume caps via Developer Console	PolicyEngine rate_limit rule	HIGH
Prompt injection	Input Guardrails + Action Gate + 4-layer security perimeter	Zero-LLM trust path (deterministic Python)	MEDIUM
Agent isolation	Firecracker microVMs per session (preventive)	OLD: Isolator tried to quarantine + freeze wallet (redundant)	HIGH
Audit records	Action Gate emits audit records (internal, not verifiable)	Ed25519 signed receipt chain (independently verifiable)	LOW

Bottom line: Leading with policy enforcement in the prize submission would make us look like we're competing with Circle's infrastructure -- and losing.

2. The Pivot: From Enforcement to Proof

'Circle protects the wallet. Verigate protects the operator.'

The pivot reframes Verigate as **complementary infrastructure**, not a competing layer. Circle enforces spending limits at the cryptographic layer. Verigate produces cryptographic proof that every decision was made correctly -- independently verifiable by third parties without trusting the operator or Circle.

Why this matters to a judge:

- **We stop competing with Circle and start building ON it** -- x401 integration, ERC-8004 reputation writes, Recibo bi-directional binding. A judge sees a team that deeply understands their stack.
- **The receipt chain is genuinely novel** -- no part of Circle's 13-component stack produces Ed25519 signed, hash-chained, Merkle-anchored, independently verifiable authorization receipts with settlement binding.
- **We solve real operator problems Circle doesn't address** -- regulatory compliance proof, forensic evidence for incident response, dispute resolution for third-party arbiters, policy state binding.
- **We fill a gap the report explicitly acknowledges** -- Circle's report says the Action Gate 'emits an audit record for compliance monitoring' but doesn't specify the format, signing, or verifiability. That gap is Verigate's entire value proposition.

3. What We Implemented

Six new capabilities were built, each integrating with Circle's ecosystem rather than duplicating it. All are live in the dashboard and golden path demo.

Feature	What It Does	Relationship to Circle
x401 Identity Binding	Issues verifiable credentials authorizing agents with scoped permissions. Executor verifies credential signature + expiry + revocation before policy eval. Credential hash is embedded in EVERY receipt (approve and deny).	Uses Circle's x401 standard. Binds it INTO receipts. Receipt now proves: WHO (x401) + WHAT policy + WHETHER approved + WHERE settled.
ERC-8004 Reputation	When the Forensic Recorder documents an incident, it publishes a REAL on-chain reputation event to the deployed AgentReputation contract on Base Sepolia. Viewable on Basescan. Other operators can query the contract.	Real deployed contract at 0xf5FE7BF0...E145AA on Base Sepolia. Every reputation event is a real on-chain tx with a clickable Basescan link.
Cross-Agent Correlation	After documenting an incident, scans all denial receipts for matching attack patterns (prompt injection, payee redirect, amount inflation, scope escape). Produces a signed correlation report: ISOLATED / SPREADING / SYSTEMIC.	Something Circle's per-session microVM isolation CANNOT do. MicroVMs prevent contamination but can't detect if multiple agents were targeted by the same attack.
Dispute Resolution	export_chain() creates a JSON file with all receipts + public key + Merkle data. verify_export() verifies everything offline. Third-party arbiters run: python -m circle.dispute verify export.json	Fills a gap Circle has NO answer for. Circle's audit records require trusting Circle's API. Verigate's export is self-contained cryptographic proof.
Recibo Binding	wallet_transfer_with_recibo() embeds the receipt hash as metadata in the on-chain USDC transfer. Creates bi-directional binding: receipt references tx AND tx references receipt.	Uses Circle's Recibo smart contract. Strong demonstration of building ON Circle's stack. Neither receipt nor settlement can exist without the other.
Offline Verifier + x401 Check	Verification pipeline now includes x401 credential binding check alongside Ed25519 signatures, hash chain, Merkle root, and anchor. All 5 checks visible in the dashboard.	Verifies that the x401 identity credential was correctly bound into each receipt. Adds a new verification dimension Circle's audit records don't have.

4. Why The Pivot Is Beneficial

4.1 Before vs After

Dimension	Before (Competing)	After (Complementary)
Headline	"Verigate stops agents from spending money they shouldn't"	"Circle protects the wallet. Verigate protects the operator."
Relationship to Circle	Duplicates Action Gate + wallet policies at app layer	Integrates with x401, ERC-8004, Recibo; fills gaps Circle doesn't address
Judge perception	"Why not just use Circle's native features?"	"This makes Circle's stack enterprise-ready for regulated operators"
Key differentiator	Policy engine (now table stakes)	Cryptographic receipt chain (genuinely novel)
Identity story	Per-tenant Ed25519 keys (primitive)	x401 credential binding (uses Circle's standard)
Incident response	Reactive quarantine + wallet freeze (overlaps with microVMs)	Forensic Recorder: signed evidence + findings + recommendations for Circle's stack
Verification	Offline verifier (4 checks)	Offline verifier (5 checks including x401) + dispute resolution CLI

4.2 Operator Problems Only Verigate Solves

Operator Need	Circle's Answer	Verigate's Answer
"Prove every payment was authorized by policy"	Transaction history from Developer Console	Ed25519 signed authorization receipts with policy hash binding
"Show what happened when our agent got compromised"	Preventive microVM isolation (no forensic evidence)	Signed forensic records with findings, evidence, and recommendations + cross-agent correlation
"A regulator/insurer needs to verify our agent's behavior"	They'd need to trust Circle's API	python -m circle.dispute verify export.json (offline, zero trust)
"Which policy was active when this decision was made?"	Policies are mutable via Console, no audit trail	Every receipt is bound to the policy_version hash
"Demonstrate EU AI Act / NIST compliance"	No automated compliance reporting	Gemini-powered compliance reports over real USDC spend data
"Flag this agent's reputation across the ecosystem"	ERC-8004 registry exists but no automated writes	Real on-chain contract deployed. Forensic Recorder auto-publishes. Tx viewable on Basescan.

5. Complementarity Matrix

How Verigate and Circle's Agent Stack work together at each layer. No overlap -- each system does what the other doesn't.

Layer	Circle's Role	Verigate's Role
Identity	x401 verifiable credentials + ZK proofs	Verify credential + bind hash into receipt chain
Policy enforcement	Action Gate + MPC co-signer (cryptographic layer)	Receipt-producing layer: proves what was decided, not enforces it
Settlement	USDC transfer on-chain via EIP-3009	Bind tx hash into signed receipt (+ Recibo reverse binding)
Audit trail	Internal audit records (not independently verifiable)	Cryptographic receipt chain (Ed25519, hash-linked, Merkle-anchored)
Incident response	Preventive microVM isolation (enforcement)	Forensic Recorder: signed evidence + findings + recommendations for Circle
Reputation	ERC-8004 registry (standard)	Real deployed contract on Base Sepolia. Auto-publish forensic events on-chain.
Compliance	Transaction history + Developer Console	Automated EU AI Act / NIST reports over real spend data
Dispute resolution	N/A	Self-contained proof chain for third-party arbiters (offline, zero trust)

Key takeaway: The old pitch made us look like we didn't understand Circle's stack. The new pitch makes us look like the missing piece that makes Circle's stack enterprise-ready for regulated operators who need **proof**, not just **protection**.

6. The Forensic Recorder (formerly Isolator)

The most important change in the pivot: the Isolator was **completely rewritten** as a Forensic Recorder. It no longer tries to enforce anything Circle already does.

Old Isolator (Removed)	New Forensic Recorder
Revokes agent identity from Verigate registry	Does NOT revoke — Circle's Action Gate handles identity enforcement
Freezes Circle wallet (calls circle wallet limit set)	Does NOT freeze — Circle's wallet policies handle spending limits
Produces an "isolation record" claiming enforcement	Produces a signed forensic record with findings and evidence
No analysis of attack vectors	Analyzes attack vector: PROMPT_INJECTION_DETECTED, UNAUTHORIZED_PAYEE, AMOUNT_VIOLATION
No recommendations	Generates actionable recommendations targeting Circle's Action Gate and wallet policies
Reputation publish was simulated (fake tx hash)	Real on-chain contract. Every event is a Basescan-verifiable transaction.

What a forensic record now contains:

- **Findings** — what happened, with evidence (e.g., 'PROMPT_INJECTION_DETECTED: Denial reasons contain adversarial instruction patterns')
- **Recommendations** — actionable items for Circle's stack (e.g., '[CIRCLE_ACTION_GATE] UPDATE_INPUT_GUARDRAILS: Review pre-LLM Input Guardrails for this attack pattern')
- **Trigger** — the denial receipt hash + reasons that triggered the analysis
- **Ed25519 signature** — independently verifiable forensic evidence

The principle: We never claim to enforce. Circle's Action Gate blocks the payment. Circle's MPC co-signer refuses to sign. Circle's microVMs isolate the agent. Verigate produces the cryptographic proof of what happened, analyzes why, and recommends what Circle's stack should do next. Two systems, zero overlap.

7. What's Live in the Dashboard

All features are implemented and visible in the live dashboard at localhost:8080. The golden path demo now runs 16 steps (up from 14) with the new capabilities.

#	Pipeline Step	What the Judge Sees
1	Wallet Check	Circle Agent Wallet funded with USDC on Base Sepolia
2	x401 Credential Issuance	Verifiable credential issued with scoped permissions, credential hash shown
3	Service Discovery	x402 services from Circle Marketplace + local endpoint
4	Gemini Ops Agent	Gemini 2.5 Flash selects service and forms payment intent
5	Verigate Gate Init	Ed25519 key + policy configured + x401 verifier attached
6	Authorized Payment	x401 verified, policy passed, USDC settled, receipt signed with identity + tx hash
7	Prompt Injection	Circle blocks it. Verigate produces signed denial receipt proving the attempt happened.
8	Forensic Recorder	Signed forensic record: findings (attack vector), evidence, recommendations for Circle's Action Gate
9	ERC-8004 Reputation	Purple card: REAL on-chain tx to deployed AgentReputation contract. Clickable Basescan link.
10	Cross-Agent Correlation	Amber card: risk assessment (SPREADING), attack patterns, recommended actions
11-16	Receipts + Merkle + Verify + Compliance	Receipt chain, Merkle tree, 5-check verification (incl. x401), compliance report

Verification panel now shows 5 checks:

- Ed25519 Signatures -- PASS
- Hash Chain -- PASS
- Merkle Root -- PASS
- x401 Identity -- PASS (new)
- Anchor -- PASS

8. On-Chain Proof: Deployed Contract + Real Transactions

The AgentReputation contract is deployed on Base Sepolia. Every reputation event from the Forensic Recorder is a real on-chain transaction. A judge can click any tx hash and verify it on Basescan.

Item	Value
Contract	AgentReputation (ERC-8004 compatible)
Address	0xf5FE7BF0163328BA0011Fa49Caf3707434E145AA
Chain	Base Sepolia (chain ID 84532)
Deploy tx	0xa80b2fe1984d947c8d85406dd44b00d691d68a6577ba7c31ae3c1b1ece606586
Test event tx	0x433dc13725e1d4631326bbac53dd7666d9fe0c0ff1d2accd7222b9f3d66e1695
Source	contracts/AgentReputation.sol (Apache-2.0)
Basescan	https://sepolia.basescan.org/address/0xf5FE7BF0163328BA0011Fa49Caf3707434E145AA

Contract interface:

- **recordEvent(agentId, eventType, severity, metadata)** — write a reputation event on-chain
- **getAgentEntryCount(agentId)** — query how many events exist for an agent
- **totalEntries()** — total events in the registry
- **ReputationRecorded** event — indexed by reporter and agentId for efficient querying

This is not a stub. Every golden path run produces a real on-chain transaction. A judge can verify it on Basescan: search the contract address, click Events, and see the ReputationRecorded logs with agent ID, severity, and forensic metadata.

9. Implementation Summary

File	Purpose	Lines
circle/x401.py	x401 credential issuance, verification, revocation -- binds agent identity into receipt chain	~180
circle/reputation.py	ERC-8004 reputation writer -- real on-chain txs to deployed contract on Base Sepolia	~130
circle/correlation.py	Cross-agent forensic correlation engine -- detects systemic attacks across agents	~200
circle/dispute.py	Standalone dispute resolution verifier CLI + PDF report generator	~250
circle/executor.py	Updated: x401 credential verification before policy eval, credential hash in all receipts	modified
circle/isolator.py	Refactored: Forensic Recorder (no enforcement). Findings, evidence, recommendations for Circle.	rewritten
circle/verifier.py	Updated: x401 binding check added to verification pipeline	modified
circle/cli.py	Updated: wallet_transfer_with_recibo() for bi-directional settlement binding	modified
circle/golden_path.py	Updated: 16 steps (was 14), x401 + reputation + correlation + dispute export	modified
app/server.py	Updated: SSE events for x401, reputation, correlation; x401 in verification state	modified
app/static/index.html	Updated: new feed cards, verification panel, overview text, defense-in-depth diagram	modified
README.md	Updated: repositioned around receipt chain, new comparison table, new components	modified

All 25 existing tests continue to pass. New modules verified with integration tests. Dashboard serves all new features at localhost:8080.