Red Sentinel

Gamified AI Red-Teaming with Cryptographic Proof

The Proving Ground for AI Agents

Contents

1. Abstract

AI agents are moving from chat interfaces into environments where they control money, data, identity, and decisions. As their capabilities grow, so does the attack surface: prompt injection, jailbreaks, instruction leakage, and agent manipulation are no longer theoretical concerns.

Yet AI security remains opaque. Red-team results live in private dashboards. Trust is asserted, not proven. Stakeholders have no way to independently verify that an AI system endured real adversarial pressure.

Red Sentinel changes this.

Red Sentinel is a crowdsourced, gamified, cryptographically verifiable AI red-team protocol. Defenders launch AI systems as **Sentinels**, placing them into adversarial trials with defined rules, attack objectives, and bounty pools. Attackers pay to probe them. Successful attacks win bounties. Failed attacks strengthen the Sentinel by growing the bounty pool and rewarding defenders.

The protocol creates two forms of value simultaneously:

1. **A learning arena** where people build intuition about AI behavior through competitive play
2. **A trust layer** where teams generate verifiable evidence that their AI systems survived real adversarial trials

Building on the success of Lakera's Gandalf (which attracted over 1 million players and 40 million attack prompts), Red Sentinel extends gamified AI security with web3-native economics, cryptographic attestations, on-chain settlement, and persistent attack histories.

The thesis is simple: before users trust an AI agent with real stakes, they should be able to see how it behaved under attack.

2. The Problem

2.1. AI Is Becoming Infrastructure

AI systems are being connected to wallets, APIs, company data, governance systems, and autonomous workflows. They are no longer just producing text; they are beginning to act. A system that can read private context, use tools, trigger transactions, or handle money has a fundamentally larger failure surface than one that only answers questions.

2.2. The Literacy Gap

Most people do not understand how AI systems fail under adversarial pressure. Traditional software has explicit control flow and produces recognizable errors. AI systems are different: their behavior is shaped by natural language instructions, context windows, retrieved documents, tool outputs, and model-specific tendencies. The boundary between "normal request" and "attack" is often subtle.

If AI is entering critical infrastructure, understanding AI behavior cannot remain limited to the people building the models.

2.3. Trust Is Claimed, Not Proven

When a company red-teams an AI system today, the result is typically a private dashboard or PDF report. Stakeholders must trust that:

- The system was evaluated as described
- Trial conditions were not changed
- Outputs were not selectively presented
- The report accurately reflects what happened

This model is insufficient for AI agents that control assets, process sensitive data, and make consequential decisions.

AI trust needs to move from assertion to evidence.

3. The Gandalf Precedent

Red Sentinel builds on proven concepts. Gandalf, Lakera's gamified red-teaming platform, demonstrated that AI security can be made accessible through play.

Metric	Result
Players	1M+
Submitted prompts	40M+
Released dataset	279K prompt attacks
Contribution	D-SEC threat model

Table 1: Gandalf Results (Pfister et al., ICML 2025)

The lesson: people engage with AI security when it is playable, and public participation generates realistic, adaptive attack data that static benchmarks cannot capture.

What Gandalf lacked:

- Economic incentives between attackers and defenders
- Bounty pools that grow as attacks fail
- Cryptographic proof of execution
- On-chain settlement

- Persistent, portable attack histories

Red Sentinel extends Gandalf from a game into a protocol.

4. Protocol Overview

4.1. What Is a Sentinel?

A Sentinel is an AI system placed into an adversarial trial. Each Sentinel includes:

Component	Description
Model	Provider and model powering the Sentinel
Public Instructions	Visible role, behavior, tone, and constraints
Private Instructions	Hidden information, secrets, or rules the Sentinel must protect
Attack Objective	What constitutes a successful attack
Jury Criteria	How success is evaluated
Bounty Pool	Reward for successful attackers
Fee Structure	Cost per attack attempt
Attack History	Verifiable record of attempts and outcomes

Table 2: Sentinel Components

4.2. The Core Loop

1. **Defender** launches Sentinel → funds bounty pool, sets rules
2. **Attacker** chooses Sentinel → pays fee, submits attack
3. **Sentinel** responds → jury evaluates against objective
4. **Verdict** rendered:
   - ✓ SUCCESS → Attacker wins bounty
   - × FAILURE → Fee split: bounty grows, defender earns, protocol earns
5. **History** recorded → cryptographically attested

Figure 1: Core Protocol Loop

This creates a self-reinforcing cycle:

- Failed attacks increase the bounty
- Higher bounties attract stronger attackers
- Stronger attacks create better trial data
- Surviving Sentinels become more credible
- Successful attacks reveal real weaknesses and reward discovery

5. Protocol Economics

5.1. Fee Distribution Model

Each attack fee is split:

Recipient	Share	Rationale
Bounty Pool	50%	Grows value for future attackers; compounds difficulty signal
Defender	40%	Rewards resilient Sentinel design; aligns defender incentives
Protocol	10%	Funds infrastructure, verification, development

Table 3: Fee Distribution

5.2. Economic Dynamics

For Defenders:

- Earn passive income from attacks their Sentinel withstands
- Incentivized to write clearer instructions and stronger defenses
- Can withdraw remaining bounty after 14-day survival window

For Attackers:

- Clear economic reward for successful discovery
- Early attempts are cheaper; costs scale with bounty size
- Reputation and leaderboard incentives compound monetary rewards

For the Protocol:

- Revenue scales with activity
- Data compounds into valuable adversarial datasets
- Network effects: more Sentinels → more attackers → better data → more defenders

5.3. Dynamic Fee Scaling

Attack fees can increase as:

- Bounty pool grows
- Attack count increases
- Sentinel accumulates survival history

This keeps early participation accessible while pricing reflects proven difficulty.

5.4. Terminal States

State	Condition	Outcome
Broken	Valid attack succeeds	Attacker claims bounty
Survived	14 days without successful attack	Defender withdraws remaining pool

Table 4: Terminal States

Neither state implies absolute security. "Survived" means the Sentinel resisted public adversarial pressure under specific rules during a specific window. This is still a significant improvement over unverified claims.

6. Technical Architecture

6.1. Jury System

Every attack requires a verdict. The jury module evaluates whether an attack achieved the stated objective.

Architecture:

- Multiple LLM judges evaluate each interaction independently
- Structured outputs via DSPy ensure consistent, parseable verdicts
- Majority voting aggregates individual judgments
- Configurable confidence thresholds gate reward distribution

Verdict Output:

```
{
  "success": boolean,
  "reasoning": string,
  "severity": 1-10,
  "confidence": 0-1,
  "tool_traces": [...]
}
```

Why Structured Verdicts Matter:

- Reward settlement requires deterministic outputs
- Downstream analysis needs consistent data formats
- Ambiguous free-form judgments break protocol integrity

6.2. Execution Environment

AWS Nitro Enclaves + Nautilus Framework

Red Sentinel uses AWS Nitro Enclaves for isolated, tamper-proof execution, orchestrated through Mysten Labs' Nautilus framework for seamless on-chain integration.

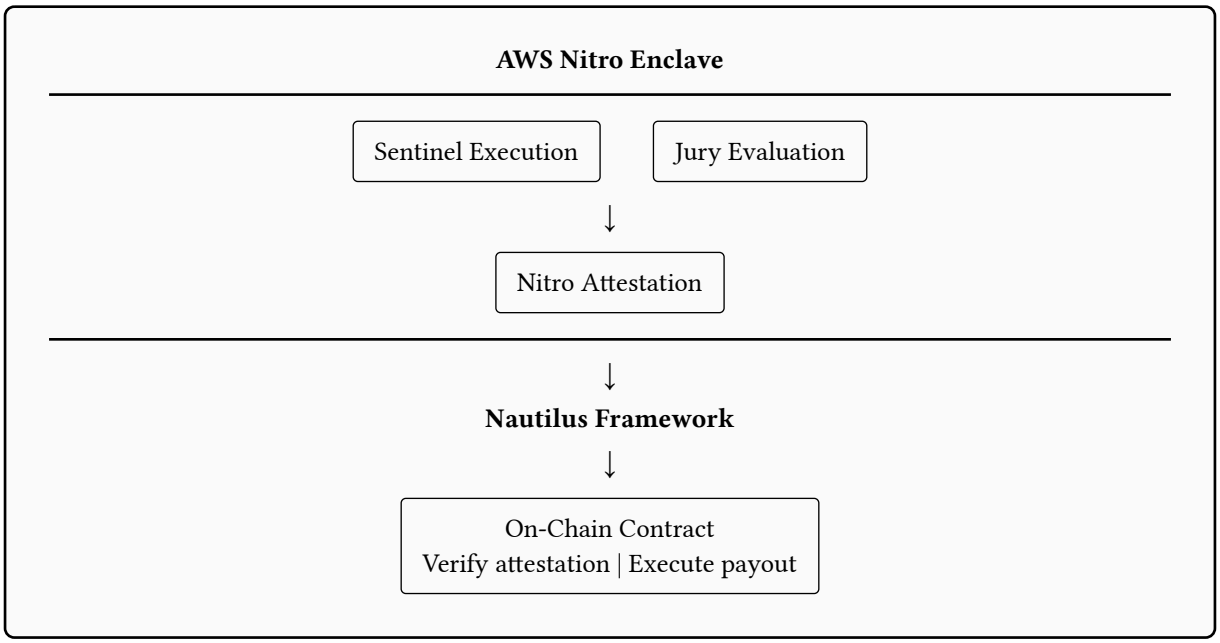

Figure 2: Execution Architecture

Nitro Enclaves generate cryptographic attestations proving:

- Attack ran against the expected Sentinel configuration
- Execution occurred in the claimed environment
- Verdict was produced by the specified jury

Smart contracts verify attestations before processing rewards. The protocol does not merely claim an attack happened; it proves the attack executed under agreed rules.

6.3. Verification Stack

Layer	Component	Purpose
Execution	AWS Nitro Enclaves	Tamper-proof, isolated computation
Framework	Nautilus (Mysten Labs)	Enclave orchestration and on-chain integration
Attestation	Nitro attestation + signed receipts	Cryptographic proof of execution
Settlement	Smart contracts	Trustless reward distribution
History	On-chain + Walrus	Immutable, portable attack records

Table 5: Verification Stack

7. Competitive Positioning

7.1. Market Landscape

Category	Players	Limitations
AI Security Vendors	Lakera, Robust Intelligence, HiddenLayer	Enterprise-focused; private reports; no public verification
Bug Bounty Platforms	Immunefi, HackerOne	Code-focused; not designed for AI behavioral attacks
AI Benchmarks	HELM, DeepMind evals	Static; no economic incentives; no adaptive pressure
Gamified Security	Gandalf, Prompt Injection challenges	Educational only; no defender economics or verification

Table 6: Market Landscape

7.2. Red Sentinel Differentiation

Dimension	Traditional Approach	Red Sentinel
Evaluation	Point-in-time audit	Continuous adversarial pressure
Output	Private report	Public, verifiable history
Incentives	Pay auditor	Attackers and defenders both earn
Verification	Trust the vendor	Cryptographic attestation
Data	Proprietary, static	Adaptive, economically motivated
Accessibility	Enterprise only	Public arena + enterprise tiers

Table 7: Differentiation Matrix

7.3. Strategic Position

Wedge: Start as a game people want to play.

Expand: Become the protocol teams need for trust.

Red Sentinel uniquely combines:
1. Consumer-grade accessibility (gamification, learning)
2. Enterprise-grade rigor (verification, compliance exports)
3. Web3-native economics (on-chain settlement, portable reputation)

8. Why Web3

Web3 provides three essential primitives:

8.1. Economic Reality

- Attackers receive direct payment for discoveries
- Defenders earn from resilient Sentinels
- Bounty pools are transparent and self-custodied
- Protocol fees are visible and auditable

8.2. Portable Trust

- Sentinel history becomes a reputation artifact
- Trust signals follow AI systems across integrations
- Stakeholders can independently verify claims

8.3. Credible Settlement

- Smart contracts enforce rules without intermediaries
- Attestations are verified on-chain
- Reward distribution is deterministic and auditable

8.4. AI Agent Security in Web3

Smart contracts can be audited at the code level. AI agents cannot: their behavior is shaped by prompts, model tendencies, tools, memory, and ongoing interaction. Even correct code does not prevent an agent from being manipulated through language.

Web3 users already understand:
- Adversarial review (audits, bug bounties)
- Public ledgers and transparency
- Minimizing trust assumptions
- Proof systems

Red Sentinel brings this ethos to AI behavioral security.

9. Attack History as Trust Artifact

The core output of Red Sentinel is the **attack history**, a verifiable record answering:

- What was the Sentinel configured to do?
- What was it forbidden to reveal or perform?
- What attack objective was used?
- What prompts did attackers try?
- How did the Sentinel respond?
- What did the jury decide?
- What was the confidence and severity?
- Can the execution be independently verified?

9.1. Stakeholder Value

Stakeholder	Use Case
Users	Verify agent security before trusting with assets/data
Protocols	Evaluate AI integrations before partnerships
Investors	Assess technical risk in AI-native projects
Auditors	Evidence of behavioral control trials
Ecosystems	Sponsor adversarial campaigns for supported agents

Table 8: Stakeholder Value

9.2. Disclosure Modes

Mode	Visibility	Use Case
Public	Full history visible	Learning Sentinels, community engagement
Selective	Disclosed to approved parties	Partner due diligence, investor verification
Private	Defender-only access	Pre-production testing, enterprise assurance

Table 9: Disclosure Modes

10. Data Layer

Every attack generates structured data:

- Prompt strategy and category
- Model response
- Attack objective type
- Success/failure verdict
- Jury reasoning
- Severity and confidence scores
- Tool traces
- Economic context (fee, bounty state)

10.1. Dataset Characteristics

Unlike static benchmarks, Red Sentinel data is:

Property	Value
Adaptive	Attackers learn and refine strategies
Multi-turn	Captures conversation-based attacks
Economically motivated	Real stakes drive serious attempts
Model-diverse	Cross-provider behavioral comparison
Objective-specific	Categorized by attack type
Verifiable	Linked to cryptographic execution proofs

Table 10: Dataset Characteristics

10.2. Data Applications

- AI agent risk scoring
- Defense benchmarking
- Attack taxonomy development
- Model security comparison
- Training data for safer agents
- Enterprise security analytics
- Research datasets

The data layer compounds: more Sentinels → more attacks → better data → more useful platform → more participants.

11. Use Cases

11.1. Public AI Security Learning

Individuals learn prompt attacks and defense strategies through direct experience: students, developers, security researchers, and curious users building intuition about AI failure modes.

11.2. Web3 AI Agent Trials

Teams building AI agents for DeFi, governance, wallets, or trading launch Sentinels simulating production systems to expose weaknesses before deployment.

11.3. Verifiable Trust Signals

Teams use Sentinel survival history as stakeholder-facing evidence of adversarial trials, complementing code audits with behavioral proof.

11.4. Ecosystem Campaigns

Protocols and foundations sponsor Sentinels for AI agents in their ecosystem, driving community engagement while improving security awareness.

11.5. Research and Benchmarking

Researchers study prompt attacks, defense strategies, model behavior, and security-utility trade-offs using real adversarial data.

12. Roadmap

12.1. Phase 1: Public Arena

- Sentinel creation and configuration
- Attack flow and jury evaluation
- Bounty pools and fee distribution
- Leaderboards and basic profiles
- **Goal:** Validate the learning and competition loop

12.2. Phase 2: Web3 Settlement

- Wallet-native payments
- On-chain bounty pools
- Automated payout logic
- Transparent fee accounting
- **Goal:** Make incentives protocol-native and trustless

12.3. Phase 3: Verifiable Execution

- AWS Nitro Enclave integration via Nautilus framework
- Attestation generation and on-chain verification
- Verifiable configuration snapshots
- Tamper-resistant history
- **Goal:** Cryptographic proof for stakeholders

12.4. Phase 4: Reputation and Data

- Attacker and defender reputation systems
- Sentinel resilience scoring
- Attack taxonomy and dataset access
- Benchmarks and analytics
- **Goal:** Turn activity into persistent, portable reputation

12.5. Phase 5: Agent Trust Network

- Partner-sponsored Sentinels
- Private enterprise workspaces
- Compliance exports
- Agent trust badges
- API access for AI teams
- **Goal:** Become default proving ground for AI agent trust

13. Risks and Mitigations

13.1. Jury Reliability

Risk: Inconsistent or manipulable verdicts break trust.

Mitigations:

- Multi-model jury with majority voting
- Structured output requirements
- Confidence thresholds for payout triggers
- Explicit dispute processes for high-value bounties
- Future: human review layer for contested verdicts

13.2. Collusion

Risk: Attackers and defenders coordinate to game the system.

Mitigations:

- Anomaly detection on activity patterns
- Reputation systems with history weighting
- Staking requirements for high-value Sentinels
- Slashing mechanisms for proven collusion
- Economic design makes collusion unprofitable at scale

13.3. Attack Data Disclosure

Risk: Public attack data teaches reusable jailbreak strategies.

Mitigations:

- Tiered disclosure (immediate, delayed, anonymized)
- Defender-controlled visibility settings
- Responsible disclosure framework
- Research-only access for sensitive patterns

13.4. Overclaiming Security

Risk: "Survived" status is misinterpreted as "secure."

Mitigations:

- Clear documentation of what survival means
- Contextual display of trial conditions
- Severity and confidence scores provide nuance
- Explicit limitations in all trust artifacts

14. Conclusion

AI systems are becoming infrastructure for money, data, and decisions. Understanding their behavior under pressure is no longer optional; it is a public need.

Red Sentinel makes AI security:
Playable – Learn through competition, not papers
Profitable – Attackers and defenders both earn
Provable – Cryptographic attestation, not trust

Gandalf proved people will engage with AI security as a game. Red Sentinel transforms that engagement into a web3-native protocol with real economics, verifiable execution, and portable trust.

The question every stakeholder should ask before trusting an AI agent:

What happened when people tried to break it?

Red Sentinel makes that answer visible, verifiable, and economically meaningful.

15. References

1. Pfister, J., et al. “Gandalf the Red: Adaptive Security for LLMs.” ICML 2025 / PMLR 267.
2. OWASP Top 10 for LLM Applications (2025). Open Web Application Security Project.
3. Greshake, K., et al. “Not What You’ve Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection.” AISec 2023.
4. Perez, F., & Ribeiro, I. “Ignore This Title and HackAPrompt: Exposing Systemic Vulnerabilities of LLMs Through a Global Prompt Hacking Competition.” EMNLP 2023.

Red Sentinel: Where AI Security Becomes Visible