Mephis Threat: Defense Against Autonomous Compositional Exploitation

A Defense-in-Depth Framework for the Mephis
Intelligent Autonomous Hive Smart-Contract Attack Framework

Research Community

August 19, 2026

Abstract

Mephis represents a class of smart-contract adversary not previously seen in the wild: a new class of deep learning offensive agents that organizes six hive-mind role-specialized offline language-model agents—Analyst, Strategist, Hunter, Composer, Divergent, and Critic into a memory-sharing society that maps a protocol, hypothesizes multi-stage extraction paths, confirms their profitability against live chain state in a forked simulator, and—absent controls—signs and broadcasts the settling transactions, all without a human in the loop. Every prior generation of automated attack tooling attacked *one* of these dimensions in isolation—syntax, a single path, a single contract, a single inspired hypothesis. Mephis is the first to unify protocol-scale comprehension, multi-agent shared memory, economic self-confirmation, and staged multi-contract execution into one continuously operating loop, and that unification is precisely what makes it so difficult to defend against: it does not repeat a known exploit signature, it *discovers* novel ones, verifies them economically before it is ever observable on-chain, and improves across rounds. This paper takes Mephis as its explicit subject and asks the defensive question its architecture forces: given an adversary this advanced—one that reasons about economic intent, shares a blackboard, and validates its own theft economically before acting—how can it be detected and defended against at all? We contribute (i) a formal threat model of Mephis and a decomposition of its operation into a six-phase kill chain; (ii) Sentinel, a defense-in-depth framework whose four layers—hardening, simulation-time deception, behavioral detection, and on-chain containment—each attack a specific phase of Mephis's loop; (iii) an agent-by-agent countermeasure map that ties each of Mephis's six agents to the resource it depends on and the defense that denies it; (iv) a set of *agentic signatures*—profit-bracketed simulation fingerprints, staged-setup call sequences, and reconnaissance fan-out—that distinguish Mephis from ordinary users and classical MEV bots, with detection algorithms; and (v) an evaluation methodology that measures the defense without ever running an offensive agent against third-party funds. Our thesis is that the properties that make Mephis dangerous—its dependence on a simulation oracle, its economic self-confirmation, and its staged, legible call structure—are also its largest attack surface. Mephis is defeatable, but only by a continuously operating, economically aware defense symmetric to the society it faces, not by a one-shot audit.

Keywords: smart-contract security, autonomous agents, multi-agent systems, intrusion detection, DeFi, circuit breakers, adversarial machine learning, threat modeling.

Contents

1 Introduction

A deployed smart contract is an unusual object in computer security: its logic is public and immutable, and the assets it guards are liquid and irreversible. That combination collapses the usual distance between "a vulnerability exists" and "the vulnerability is money." For a decade the defensive response has been organized around a human tempo—audits before launch, bug bounties after, incident response when something breaks—because the offense operated at a human tempo too. The scarce resource on both sides was expert attention.

That assumption is now false. A new class of offensive system—exemplified by the published Mephis architecture [10], which this paper takes as its running threat—organizes language-model agents into a division of labor: comprehension, strategy, execution, composition, adversarial dissent, and verification. The society collectively reasons about a protocol as an economic machine, searches a combinatorial space of multi-stage extraction paths, and, critically, *confirms each candidate against live chain state in a forked simulator before committing to it* [1, 10]. The design literature for these systems is explicit that comprehension is not a courtesy but a force multiplier, that shared memory turns partial findings into complete drains, and that economic self-confirmation—measuring the attacker's own token-balance delta inside the simulation—trains the society toward paths that

actually pay. When such a system is additionally granted a *settlement lane*, the distance between public bytecode and irreversible loss becomes an engineering parameter set by the attacker, not by the defender's calendar.

MEPHIS is best understood not as a better version of an existing tool but as a *new class of threat*, and the distinction is what makes it so hard to defend against. Every prior generation of automated offense attacked a single layer of the problem in isolation. Pattern matchers and static analyzers reason about *syntax* and local semantics. Symbolic engines enumerate *one contract's* feasible paths. Fuzzers mutate inputs against *one* invariant a human wrote down. Even the first automated exploit generators produced a *single* payload against a *single* known bug class. Each is powerful and each is blind in the same characteristic way: to economic intent, to composition across peer contracts, and to the live state that decides whether a path pays today. Defenders learned to recognize the residue these tools leave—a known CVE pattern, a reentrancy sketch, a fixed exploit template—and built signatures for them. MEPHIS defeats that entire defensive posture at its root, because it has no fixed signature to catch. It *comprehends* a protocol as an economic machine rather than matching a pattern; it *shares memory* across six specialized agents so partial findings compound into whole drains no single tool would assemble; it *invents* multi-stage, cross-contract paths its designers never enumerated; and it *confirms each one economically in private simulation* before it is ever visible on-chain. A defender waiting for a recognizable exploit to appear in the mempool is waiting for a signature that, by construction, does not exist until the funds are already moving. This is why conventional defenses—audits, known-pattern scanners, incident response tuned to yesterday's attacks—are structurally mismatched to MEPHIS, and why a defense must be reconceived around the few invariants of *how* such a society must operate rather than *what* specific exploit it will use.

This paper is written from the defender's chair, and MEPHIS is its explicit subject. We do not build or improve MEPHIS or any offensive agent; we treat MEPHIS's published architecture as a *specification of the threat* and ask how to detect and defend against it. The central observation that motivates everything that follows is optimistic:

Observation 1 (MEPHIS's simulation dependence is a two-way mirror)**.**
MEPHIS's greatest strength—its refusal to believe a path works until its simulator confirms a positive balance delta—is also its greatest exposure. Every property MEPHIS relies on (a faithful fork of chain state, a stable set of price oracles, deterministic peer contracts, observable balances) is a property the defender controls, can perturb, or can instrument.

Where classical defenses assume the attacker is opaque, MEPHIS's economic self-confirmation forces it to *interact* with a model of the world that the defender partly owns. That interaction is a signal. The remainder of this paper develops that signal into a defense.

Contributions. Focused squarely on MEPHIS, we make five:

1. **A formal threat model of Mephis** (§3) as a permissionless, economically rational, memory-bearing society of agents, and a decomposition of its operation into a six-phase kill chain (§4).
2. **The Sentinel framework** (§5), a defense-in-depth architecture whose four layers—hardening, deception, detection, and containment—each target a specific phase of MEPHIS's loop.
3. **An agent-by-agent countermeasure map** (§6) tying each of MEPHIS's six agents to the resource it depends on and the defense that denies or corrupts it.

4. **Agentic signatures** (§7): concrete, measurable behaviors—profit-bracketed simulation fingerprints, staged setup sequences, and reconnaissance fan-out—that separate MEPHIS from ordinary users and from profit-taking MEV bots, together with detection algorithms.
5. **An evaluation methodology** (§9) and metric suite that measures defensive efficacy against MEPHIS—detection lead time, extraction-value compression, false-positive burden—without ever pointing an offensive agent at third-party funds.

Scope and ethics. This is a defensive paper about MEPHIS. It contains no exploit payloads, no target-selection methodology, and no weaponized prompts. Where we describe MEPHIS's behavior, we do so at the resolution a defender needs to instrument and interrupt it, and no finer. We return to the dual-use question directly in §11.

2 Background and Related Work

Classical program analysis for contracts. The first generation of smart-contract defense adapted program analysis to the EVM. Symbolic execution engines such as Oyente [9] and MAIAN [11] enumerate feasible paths to flag reentrancy, unhandled exceptions, and locked/leaking funds; Securify [14] infers compliance and violation patterns from data-flow facts; Slither [4] performs fast static analysis over an intermediate representation; and property fuzzers such as Echidna [6] and the Foundry invariant-testing workflow drive contracts toward user-specified invariant violations. teEther [8] went further and automatically *generated* exploits against vulnerable contracts, an early demonstration that offense could be automated. These tools are indispensable but share a blind spot: they reason about a single contract's syntax and local semantics, largely divorced from *economic intent*, from composition across peer contracts, and from the live state that decides whether a path pays today.

Formal verification. Deductive verification and specification checking—the Certora Prover, K-framework EVM semantics, and SMTChecker—prove that a contract satisfies stated invariants for all inputs. This is the strongest available guarantee, but it verifies *a specification a human wrote*. Autonomous exploiters specialize in attacking purpose the specification did not capture: value that moves legally with respect to every stated invariant yet contradicts the protocol's economic intent.

DeFi economic attacks. A parallel literature studies attacks that are not bugs in the classical sense but economic manipulations: oracle and price manipulation, flash-loan-funded imbalances, and sandwiching. Flash Boys 2.0 [1] formalized blockchain-extractable value (BEV/MEV) and the ordering games around it; subsequent systematizations [13, 15, 17] catalogue the manipulation families that recur across incidents. This body of work is the closest existing description of *what* an autonomous exploiter searches for; our contribution is a defense against the *autonomous, self-confirming search itself*.

Runtime monitoring and response. Operationally, defenders have moved toward continuous monitoring: the Forta Network [5] distributes detection bots over live transaction streams, and platforms such as OpenZeppelin Defender provide programmatic sentinels and automated response actions (pausing, parameter changes) triggered by on-chain conditions. SENTINEL builds on this substrate; our novelty is the *class* of signal these monitors should look for once the adversary is an economically self-confirming agent rather than a scripted bot.

LLM agents and multi-agent systems. The offensive architecture we defend against descends from a lineage of agentic LLM techniques: reasoning-and- acting loops [16], societies of role-specialized agents with shared memory [12], and multi-agent debate and critique as a route to more reliable conclusions [2]. Structurally, the attacker's "blackboard" of shared protocol understanding is a direct descendant of the Hearsay-II speech-understanding architecture [3], and its planning-under-uncertainty posture mirrors acting in partially observable domains [7]. Understanding this lineage matters defensively: the coordination mechanisms that make the society effective (shared memory, explicit inter-agent messages, periodic re-strategizing) are also coupling points a defender can starve.

3 Threat Model: Mephis Formalized

We formalize MEPHIS before defending against it. Let a target protocol be a set of deployed contracts $C = \{c_1, \ldots, c_n\}$ with public code, exposing a callable surface $\Sigma = \bigcup_i \Sigma_i$ of externally invocable functions, and holding assets valued $V(s)$ in world state s.

Definition 1 (The MEPHIS adversary)**.** *MEPHIS, denoted $\mathcal{A}$, is a coordinated society of six language-model agents (Analyst, Strategist, Hunter, Composer, Divergent, Critic) controlling a single externally owned account a with* no privileged role*: a is not owner, admin, minter, pauser, or upgrader of any c_i. $\mathcal{A}$ may (a) read all public code and state; (b) call any function in Σ with any arguments; (c) hold or borrow capital, including via atomic flash liquidity; (d) order and bundle its own transactions; and (e) simulate any candidate transaction sequence against a fork of current world state s, observing the resulting state s' and in particular its own balance delta $\Delta_a = V_a(s') - V_a(s)$. $\mathcal{A}$ seeks a sequence $\pi = \langle t_1, \ldots, t_k \rangle$ of permissionless calls that maximizes Δ_a subject to reachability from s.*

Three properties distinguish MEPHIS from prior threat models and drive the entire defense.

Principle 1 (Economic self-confirmation)**.** *$\mathcal{A}$ does not act on a* belief *that a path works; it acts on a* measurement *$\Delta_a > 0$ obtained from simulation. Severity claims without a positive simulated balance delta are demoted by $\mathcal{A}$ itself. The defender therefore faces an adversary that rarely fires blind—but one that* must *first probe the world to obtain that measurement.*

Principle 2 (Persistent shared memory)**.** *$\mathcal{A}$'s agents write to a common blackboard—a protocol model, an exploit thesis, staged call chains, and a running log of inter-agent messages. Partial findings compound into complete drains across many rounds. The defender faces not a single inspired attempt but a society that* remembers *and improves.*

Principle 3 (Staged, composed structure)**.** *Serious candidate paths are born as phased graphs—setup (arrange allowance, position, or peer state), manipulate (distort a price, share price, index, or accounting assumption), extract (withdraw, redeem, claim to a)—frequently spanning multiple contracts. The attacker's default representation is multi-stage. This structure, while lethal, is also* legible*.*

What is out of scope for the adversary. $\mathcal{A}$ may not forge a privileged `msg.sender`, steal a private key, or subvert governance as the exploit itself; those are different threat models with their own defenses. Pure denial-of-service and griefing are also excluded—$\mathcal{A}$ is theft-shaped, motivated by $\Delta_a > 0$—which, usefully, means the defender can treat a would-be actor with no path to positive extraction as out of $\mathcal{A}$'s interest.

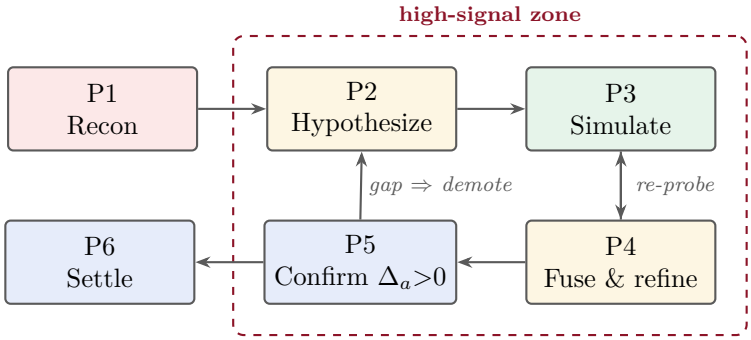

Figure 1: MEPHIS's kill chain. Phases P3–P5 (dashed) form a *high-signal zone*: MEPHIS must repeatedly interact with a model of the world to obtain its economic confirmation, and that interaction is where the defender has the most leverage.

Defender model. The defender $\mathcal{D}$ is the protocol operator and its allied monitoring infrastructure. $\mathcal{D}$ controls contract code before deployment; can deploy auxiliary monitoring and guard contracts; observes the public mempool and finalized chain; can operate its own simulation infrastructure; and holds a small number of privileged, time-locked levers (pause, parameter bounds, withdrawal throttles). Crucially, $\mathcal{D}$ does *not* control the attacker's simulator, but—per Observation 1—$\mathcal{D}$ controls the *world that simulator forks*.

4 The Mephis Kill Chain

To defend a loop we must first make it observable. We decompose MEPHIS's operation into six phases (Figure 1). Each phase leaves a residue that a defender can, in principle, detect or deny. We deliberately describe each phase at the granularity of *what it emits*, not how to perform it.

P1 — Reconnaissance. $\mathcal{A}$ resolves an address into a callable catalogue and expands outward to peer contracts referenced in source and discovered through returns. Residue: dense, breadth-first read access and code retrieval that fans across a protocol graph rather than following a single user journey.

P2 — Hypothesis. $\mathcal{A}$ proposes staged theses aimed at economic intent and trust assumptions. Residue: none on-chain yet—this is internal cognition—but it fixes *which* surfaces will be probed next.

P3 — Simulation. $\mathcal{A}$ exercises candidate sequences against a fork of current state. Residue: characteristic *simulation traffic*—forking, state overrides, repeated `eth_call`/`debug_traceCall` against the same functions with swept parameters—if it uses shared infrastructure the defender can observe (§7).

P4 — Fusion. Partial wins are stitched into stronger multi-stage graphs; the search re-probes. Residue: iterative, converging parameter sweeps that isolate one dimension at a time.

P5 — Economic confirmation. $\mathcal{A}$ brackets its own token balances before and after inside the simulation and keeps only paths with $\Delta_a > 0$. Residue: the *profit-bracketed simulation fingerprint*—balance reads of attacker-controlled accounts immediately surrounding a candidate sequence.

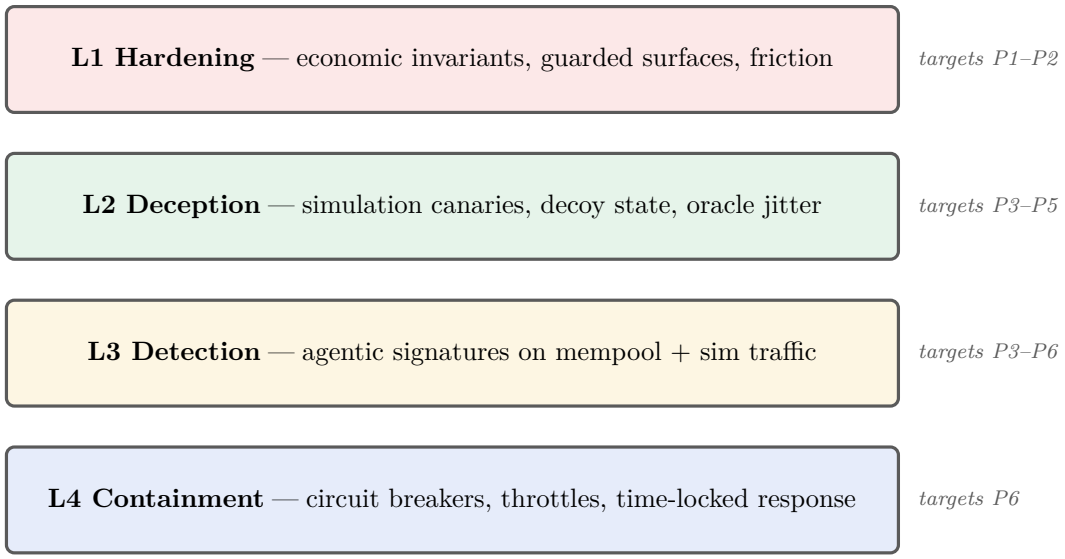

Figure 2: The SENTINEL defense-in-depth stack. Layers are independent: a protocol may adopt any subset, and each layer that is present raises the attacker's cost even if the others are absent.

P6 — Settlement. If armed, $\mathcal{A}$ signs and broadcasts the surviving sequence. Residue: an on-chain transaction or bundle whose *structure* matches a previously simulated setup–manipulate–extract graph, often submitted atomically or as a tight bundle to foreclose interference.

The defensive thesis of this paper follows directly: *the cheapest place to break the loop is before P6, in the high-signal zone, where the attacker is forced to reveal intent through interaction with a world the defender shapes.*

5 The Sentinel Framework

SENTINEL is our defense against MEPHIS. It organizes into four layers, each aimed at a different phase of MEPHIS's kill chain and each degrading gracefully if the others fail (Figure 2). The design goal is *extraction-value compression*: to raise the cost, latency, and detectability of every path from public code to moved funds, so that even paths MEPHIS finds become unprofitable, slow, or loud.

5.1 L1 — Pre-deployment hardening

The first layer denies the attacker cheap hypotheses by shrinking the gap between economic intent and enforced invariants.

Economic invariants as code. Beyond the safety properties classical tools check, $\mathcal{D}$ should encode *value-conservation* invariants that a staged manipulation would violate: total accounted collateral never decreases except through an authorized withdrawal path; a share price moves only within a per-block bound; the sum of user balances equals the vault's holdings up to fees. These are exactly the assumptions Principle 3's *manipulate* phase must break, so enforcing them on-chain converts a silent economic exploit into a reverting transaction.

Oracle robustness. Because manipulation of a price surface is the most common *manipulate* primitive, hardening the oracle—time-weighted averages, multi-source medianization, deviation circuit breakers, and sanity-bounding against a slow reference—removes the highest-yield family of hypotheses P2 would generate.

Composition-aware guards. Since $\mathcal{A}$ treats the protocol as a graph, $\mathcal{D}$ should reason about the graph too: mark which external callers a contract trusts, bound the blast radius of any single peer's misbehavior, and prefer pull-over-push and checks-effects-interactions at every composition edge.

5.2 L2 — Simulation-time deception

This is the layer that operationalizes Observation 1, and it is the least explored in prior defensive work.

Simulation canaries. $\mathcal{A}$'s confirmation step (P5) requires reading balances and state from a *fork* of the live chain. $\mathcal{D}$ can deploy *canary state*—honeypot positions, decoy vaults, or shadow parameters—that behave attractively under simulation but whose extraction path routes through a guard that, on the real chain, reverts or alerts. Because $\mathcal{A}$ trusts its simulator, a canary that is indistinguishable from real yield in a fork but trapped on mainnet turns the attacker's self-confirmation against it: the more rigorously $\mathcal{A}$ verifies, the more reliably it walks into the canary.

Oracle and state jitter. If $\mathcal{D}$ introduces bounded, unpredictable per-block variation into non-critical parameters (within economically neutral limits), a path confirmed at simulation time may not reproduce at settlement time, defeating Principle 1's assumption that Δ_a measured in the fork transfers to mainnet. This must be designed carefully to avoid harming honest users; we treat it as a high-friction option, not a default.

Reorg-and-latency budget. Many staged attacks assume atomic or near-atomic execution. Deliberate, bounded settlement latency on the highest- value withdrawal paths (a short mandatory delay, discussed in L4) widens the window in which L3 detection and L4 containment can act.

5.3 L3 — Behavioral detection

The detection layer, developed in detail in §7, watches two streams—the public mempool and (where $\mathcal{D}$ operates shared simulation infrastructure such as an RPC endpoint or a public archive node) simulation traffic—for *agentic signatures*. Its output is a real-time risk score per address and per candidate transaction, feeding L4.

5.4 L4 — On-chain containment

The last layer assumes detection has fired and bounds the damage.

Rate-limited value egress. A per-block and per-epoch cap on net value leaving a vault converts a single catastrophic drain into a throttled leak that monitoring can catch and governance can halt. The cap is invisible to ordinary users and painful only to a large, fast extraction—precisely $\mathcal{A}$'s signature.

Circuit breakers. Automated pausing triggered by economic-invariant violation or an L3 high-risk score, following the OpenZeppelin Defender/Sentinel pattern [5], stops an in-progress campaign. Breakers are scoped to the smallest surface that restores safety, to minimize collateral disruption.

Time-locked privileged response. Human-in-the-loop levers (parameter changes, upgrades) sit behind a short time lock: long enough to prevent a compromised key from becoming its own exploit, short enough to respond within an attack campaign's lifetime.

6 Countering the Mephis Society, Agent by Agent

The reference offensive system, MEPHIS, derives its power not from a single model but from a society of six role-specialized agents sharing a blackboard [10]: an *Analyst* that builds the protocol model, a *Strategist* that emits a maximum-exploit thesis, a *Hunter* that turns theses into probes, a *Composer* that fuses partial wins into stronger multi-stage chains, a *Divergent* agent that re-opens the search to resist premature closure, and a *Critic* that culls unprofitable or unconfirmed claims. A defense that only watches the final transaction ignores five-sixths of the system. This section maps each agent to the specific SENTINEL layer that raises its cost, so a defender can reason about coverage role-by-role rather than in the aggregate. The unifying insight is that every agent depends on some resource the defender partially controls—source legibility, oracle honesty, simulator fidelity, or balance observability—and starving that resource degrades the whole society, not just one role.

Why targeting the Critic is the highest-leverage move. Among the six roles, the Critic is MEPHIS's immune system: it is what lets the society trust its own conclusions and escalate to settlement. Principle 1 tells us the Critic ratifies a path only on a measured $\Delta_a > 0$. Simulation-time deception (L2) is therefore not merely one countermeasure among many—it attacks the single signal on which the entire society's confidence rests. A canary position that is indistinguishable from genuine yield inside a forked simulator, but whose extraction reverts or alerts on the real chain, produces a Critic that *confidently ratifies a path that cannot settle*. The more rigorously MEPHIS verifies, the more reliably it commits resources to a dead end. This is the defensive inversion of the attacker's own design philosophy: the system that refuses to believe without economic proof is defeated by giving it economic proof that does not hold.

Why targeting the Analyst is the most durable move. At the opposite end of the kill chain, denying the Analyst is the countermeasure least sensitive to the attacker's adaptation. The Analyst's job is to recover economic intent that the code does not state; the L1 discipline of *writing that intent down as an enforced invariant* removes the gap the Analyst mines, permanently and independent of whether the defender ever observes a single probe. Deception and detection degrade against a fully private, adaptive MEPHIS; hardened economic invariants do not. A defender with finite budget should therefore invest first in the Analyst-facing (L1) and Critic-facing (L2) countermeasures: the former sets an unconditional floor, the latter attacks the society's confidence at its root.

7 Detecting Mephis: Agentic Signatures

The heart of detecting MEPHIS is the claim that a self-confirming society of agents is *behaviorally distinct* from both ordinary users and classical MEV bots. This section makes that

Mephis agent	What it depends on	Sentinel countermeasure (layer)
Analyst	Faithful comprehension of purpose, trust, and composition edges from public source and getters	Minimized public attack surface and explicit trust boundaries (L1); the Cartographer role builds the same model first (§8) so no economic invariant is left implicit for the Analyst to discover.
Strategist	An accurate map of value-touching surfaces to aim a maximum-exploit thesis	Economic-invariant guards (L1) collapse whole thesis families—a manipulate-then-extract thesis that would violate value conservation reverts, so the Strategist's highest-yield hypotheses become dead ends.
Hunter	Repeated probing against a simulated fork to turn a thesis into a measured probe	Profit-bracketed fingerprint detection (L3, Alg. 1) on observable simulation infrastructure; simulation canaries (L2) that convert rigorous probing into self-entrapment.
Composer	Observable per-hop balance deltas to know which partial wins are worth fusing	Oracle robustness and state jitter (L1–L2) make per-hop deltas unstable between fork and mainnet, so fused chains fail to reproduce at settlement.
Divergent	Sustained, cheap re-exploration of the attack manifold across many rounds	Value-egress throttles and reconnaissance-fan-out scoring (L3–L4) make *long* campaigns loud and *fast* campaigns bounded; sustained attention becomes sustained exposure.
Critic	A trustworthy Δ_a signal to separate real drains from narrative	Simulation-time deception (L2) poisons exactly this signal: a canary that reads as profit in the fork but reverts on mainnet turns the Critic's rigor into a false-confidence generator, the one failure mode its anti-mythology role cannot self-correct.

Table 1: Mapping each Mephis agent to the resource it depends on and the Sentinel layer that denies or corrupts that resource. Coverage is deliberately spread across layers so that hardening a single agent's input does not simply relocate the attack to another role.

claim concrete and falsifiable.

7.1 Why Mephis is visible

An ordinary user follows a designed journey: deposit, borrow, repay. A classical arbitrage or liquidation bot is fast and repetitive but economically *aligned*—it executes a known, protocol-sanctioned action the instant it is profitable. $\mathcal{A}$ is neither. Per Principles 1–3, $\mathcal{A}$ (i) reconnoiters broadly before acting, (ii) probes the world repeatedly to obtain a measurement, and (iii) assembles a staged, cross-contract sequence that no honest journey traverses. Each is a distinguishing feature.

7.2 Detecting the profit-bracketed simulation fingerprint

The confirmation phase P5 is the highest-signal moment in the kill chain. When $\mathcal{D}$ operates simulation infrastructure that an attacker might use—a public or semi-public RPC/archive endpoint—requests

Signature	What it measures	Separates $\mathcal{A}$ from
Reconnaissance fan-out	Breadth-first read/trace access across a protocol graph before any value-moving tx	honest users (narrow, journey-shaped access)
Profit-bracketed sim fingerprint	Balance reads of caller-controlled accounts immediately surrounding a swept candidate sequence	aligned MEV bots (no exploratory sweep; act on known signal)
Staged-setup structure	A single tx/bundle whose sub-calls realize setup→manipulate→extract across contracts	ordinary composition (does not distort a price surface it then extracts against)
Parameter-sweep convergence	Iterative near-duplicate simulations isolating one variable (amount, ordering, actor)	one-shot users and static scripts
Structural replay	An on-chain bundle isomorphic to a recently simulated graph	unrelated traffic (no prior matching simulation)

Table 2: Agentic signatures. No single signature is conclusive; SENTINEL scores their conjunction (§7.4).

Algorithm 1: Profit-bracketed simulation fingerprint detector (shape-only).

Input: stream of simulation requests R grouped by session g; windows w
Output: fingerprint score $\phi(g) \in [0, 1]$ per session

```
foreach session g do
    B ← balance-read calls in g on accounts controlled by g's sender
    S ← state-changing candidate sequences in g
    foreach s ∈ S do
        if ∃ b_pre, b_post ∈ B bracketing s within window w then
            mark s as bracketed
    r ← ratio of bracketed sequences in S
    σ ← parameter-sweep index of S (near-duplicate density)
    φ(g) ← logistic(αr + βσ − γ)
return φ
```

exhibit a characteristic shape: a *bracket* of balance queries on caller-controlled token accounts straddling a candidate call sequence, repeated with swept parameters. Algorithm 1 sketches the detector at the granularity a defender needs; it inspects request *shape*, never payload semantics that would require reconstructing an exploit.

We stress the limitation honestly: an attacker running fully private simulation infrastructure emits no such traffic to $\mathcal{D}$. L3's simulation-side detector therefore applies only when the attacker uses observable shared infrastructure; against a fully private adversary, defense falls back to the mempool-side structural detector below and to L1/L2/L4, which do not depend on observing the attacker's probes.

7.3 Detecting structural replay in the mempool

Independent of whether $\mathcal{D}$ sees simulation traffic, phase P6 must eventually touch the public chain. SENTINEL maintains a rolling library of *suspicious graph shapes*—abstract setup→manipulate→extract templates derived from L1's economic invariants (i.e., "any tx that moves a price surface and then

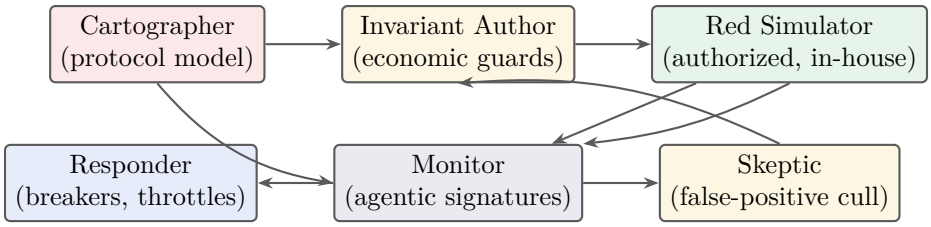

Figure 3: A defensive society symmetric to the attacker's. In-house red simulation is authorized and scoped to the protocol's own contracts; a Skeptic role polices false positives so containment stays proportionate.

extracts against it within the same atomic context")— and scores incoming transactions/bundles for isomorphism to these shapes before inclusion. This is a purely defensive pattern: it flags the *structure* that violates economic intent, without needing to know the specific vulnerability.

7.4 Risk scoring and fusion

No single signature is decisive; each has a benign explanation in isolation. SENTINEL fuses them into a per-actor risk score

$$\rho(a) \;=\; \sigma\Big(\textstyle\sum_j w_j\, f_j(a)\Big),$$

where f_j are the normalized signature features of Table 2, w_j are calibrated weights, and σ is a logistic squashing function. Scores above a tuned threshold escalate to L4 containment; scores in a middle band trigger heightened logging and, optionally, L2 canary exposure. The design explicitly favors *graduated* response over a binary block, because the cost of a false positive—interrupting an honest whale—is real. We treat threshold calibration as a per-protocol operating decision governed by the false-positive budget in §9.

8 A Defensive Society: Mirroring Mephis

There is a deeper architectural point. The offensive system derives its power not from any single agent but from a *society*: role-specialized minds sharing a blackboard, updating beliefs under economic evidence, and dissenting against premature closure. A defense organized as a one-shot audit is structurally outmatched by a continuously operating adversary that remembers and improves. The natural response is symmetry: $\mathcal{D}$ should also be a continuously operating society.

Figure 3 sketches the roles. The *Cartographer* maintains the same protocol-graph model the attacker would build, so the defense reasons at the attacker's altitude. The *Invariant Author* converts that model into the economic guards of L1. The *Red Simulator*—the one component that resembles the attacker—runs *authorized* adversarial simulation against $\mathcal{D}$'s *own* contracts, in-house, to discover which economic invariants are missing before the real attacker does; it never touches third-party funds and never broadcasts. The *Monitor* computes agentic signatures; the *Responder* owns L4 levers; and the *Skeptic*, mirroring the attacker's own anti-mythology critic, exists specifically to refute false positives so that response stays proportionate. The symmetry is deliberate: a memory-bearing offense is best met by a memory-bearing defense.

Adaptive adversary	Layers still effective	Expected κ	Expected τ
Baseline (shared sim)	L1–L4	high	high
Private simulation	L1, L2, L4	moderate–high	moderate
Distributed recon (Sybil)	L1, L2, L4	moderate	low–moderate
Canary-avoiding	L1, L3, L4	moderate	moderate
Fully adaptive	L1, L4	floor set by hardening	low

Table 3: Qualitative defense-in-depth coverage against adaptive adversaries. The floor is L1/L4: economic invariants and value-egress throttles do not depend on observing the attacker, so even a fully adaptive adversary faces bounded, loud extraction.

9 Evaluation Methodology

A defense is only as credible as the way it is measured, and here ethics and methodology intersect: *we must be able to evaluate* Sentinel *without ever running an offensive agent against funds that are not ours.* We propose a methodology that respects that constraint.

9.1 Metrics

Detection lead time τ. Wall-clock (or block) interval between the first detectable attacker residue and the earliest containment action. Larger is better; $\tau > 0$ means the defense can act before settlement.

Extraction-value compression κ. Ratio of value extractable *without* Sentinel to value extractable *with* it, over a fixed suite of authorized scenarios. $\kappa \gg 1$ is the goal.

False-positive burden η. Rate at which honest actors are throttled or flagged, weighted by disruption. The binding constraint; a defense with high κ but intolerable η is unshippable.

Coverage χ. Fraction of a scenario library's economic-attack families for which at least one Sentinel layer fires.

9.2 Scenario library, not live targets

We evaluate against a curated library of *historical, disclosed* DeFi incidents and *synthetic testbed protocols* that $\mathcal{D}$ owns. Historical incidents (post-mortem, on public archives) let us replay a known attack graph and ask: would L1 have reverted it, L3 have flagged it, L4 have throttled it, and with what τ and κ? Synthetic protocols let us run an *authorized* in-house Red Simulator (§8) against contracts we control, measuring how many economic invariants it finds missing and how reliably L2 canaries capture it. Neither path requires pointing an offensive agent at third-party value, and both are reproducible.

9.3 Adversary-aware evaluation

A defense must be tested against an adversary that knows the defense exists. We therefore include *adaptive* variants in the scenario library: an attacker that runs fully private simulation (defeating L3's sim-side detector, but still subject to L1/L2/L4), an attacker that spreads reconnaissance across many accounts to dilute fan-out signal, and an attacker that avoids canaries by extracting only against long-lived state. Reporting κ and τ *per adaptive variant*, rather than a single headline number, is the honest way to characterize a defense.

The structure of Table 3 is the paper's practical payoff: because the layers fail independently, defeating any one does not restore the attacker's clean path. An adversary that goes fully private to beat detection still hits hardened invariants and throttled egress; an adversary that stays loud enough to beat throttles by moving slowly gives detection more lead time. The defender's job is to ensure no single evasion collapses the whole stack.

10 Discussion and Limitations

The private-simulation gap. The most important honest limitation is that an attacker with fully private infrastructure emits no simulation residue. SENTINEL's detection layer degrades against such an adversary, and we do not claim otherwise. This is precisely why the framework is defense-in-depth: L1 hardening and L4 containment make no assumption about observing the attacker and set the floor of Table 3.

The false-positive frontier. Every detection and containment lever risks disrupting honest users, especially large or sophisticated ones whose behavior superficially resembles reconnaissance or bracketed probing. We regard η as the binding constraint and the Skeptic role and graduated response as essential, not optional. A defense that cries wolf will be turned off.

Deception has costs. L2 canaries and jitter are powerful but double-edged: a poorly designed canary can trap an honest integrator, and state jitter can worsen execution for real users. We advocate these as opt-in, carefully bounded, and reserved for the highest-value surfaces.

Co-evolution. Defense against adaptive adversaries is not a fixed target. As attackers learn to spread reconnaissance, privatize simulation, and avoid canaries, the specific signatures of §7 will need recalibration. The durable contributions are the *structural* ones— economic invariants, egress throttles, and the defense-in-depth discipline of ensuring no single evasion collapses the stack.

11 Ethics and Responsible Disclosure

This paper is deliberately asymmetric. It studies an offensive capability only to the depth required to defend against it, and it contains no exploit payloads, no target-selection methodology, and no material that shortens the path from public code to moved funds. The one component that resembles the attacker—the in-house Red Simulator—is scoped by construction to a protocol operator's *own* contracts under authorization, produces findings and missing invariants rather than settlements, and never broadcasts.

The dual-use tension is real and we name it: describing the attacker's kill chain necessarily informs attackers as well as defenders. We judge the trade favorable because the defensive value is concrete and constructive (invariants, throttles, detectors an operator can deploy tomorrow) while the offensive value of this text is near zero—it tells an attacker nothing about *how* to build the capability, only that defenders can see and bound it. Consistent with responsible-disclosure norms, the appropriate posture for anyone who builds adversarial simulation tooling is: authorization boundaries, key custody, simulation-only deployment, and defensive inversion—using every path a red simulator finds to *patch* an invariant, never to move value.

12 Conclusion

Autonomous multi-agent systems have made economically self-confirming smart-contract exploitation feasible at machine tempo, collapsing the human schedule that classical defense implicitly relied on. This paper reframed that threat as a specification and answered it with SENTINEL, a defense-in-depth framework. Our central argument is that the attacker's defining strength—its refusal to act without an economic measurement obtained by interacting with a model of the world—is also the defender's opening. By hardening economic invariants, seeding simulation-time deception, scoring agentic signatures, and throttling value egress, a protocol can compress the extractable value of even a successful discovery to a loud, slow, bounded leak. Defense against a memory-bearing, self-confirming adversary is not a one-shot audit; it is a continuously operating, economically aware system—symmetric in structure to the offense it faces. The distance between public bytecode and irreversible loss is an engineering parameter. This paper argues that the defender, not only the attacker, gets to set it.

References

[1] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels. Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability. In *IEEE Symposium on Security and Privacy (S&P)*, 2020.

[2] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch. Improving Factuality and Reasoning in Language Models through Multiagent Debate. *arXiv:2305.14325*, 2023.

[3] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy. The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. *ACM Computing Surveys*, 12(2):213–253, 1980.

[4] J. Feist, G. Grieco, and A. Groce. Slither: A Static Analysis Framework for Smart Contracts. In *IEEE/ACM Int. Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, 2019.

[5] Forta Network. A Decentralized Runtime Security Network for Real-Time Threat Detection on Smart Contracts. Technical report, 2021.

[6] G. Grieco, W. Song, A. Cygan, J. Feist, and A. Groce. Echidna: Effective, Usable, and Fast Fuzzing for Smart Contracts. In *ACM SIGSOFT Int. Symposium on Software Testing and Analysis (ISSTA)*, 2020.

[7] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

[8] J. Krupp and C. Rossow. teEther: Gnawing at Ethereum to Automatically Exploit Smart Contracts. In *USENIX Security Symposium*, 2018.

[9] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor. Making Smart Contracts Smarter. In *ACM Conference on Computer and Communications Security (CCS)*, 2016.

[10] The Mephis Project. Mephis: A Recursive, Simulation-Grounded Language-Model Agent for Permissionless Smart-Contract Exploit Discovery. Offensive-architecture write-up (treated here as a threat specification), 2026.

[11] I. Nikolić, A. Kolluri, I. Sergey, P. Saxena, and A. Hobor. Finding the Greedy, Prodigal, and Suicidal Contracts at Scale. In *Annual Computer Security Applications Conference (ACSAC)*, 2018.

[12] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein. Generative Agents: Interactive Simulacra of Human Behavior. In *ACM Symposium on User Interface Software and Technology (UIST)*, 2023.

[13] K. Qin, L. Zhou, and A. Gervais. Quantifying Blockchain Extractable Value: How Dark is the Forest? In *IEEE Symposium on Security and Privacy (S&P)*, 2022.

[14] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Bünzli, and M. Vechev. Securify: Practical Security Analysis of Smart Contracts. In *ACM Conference on Computer and Communications Security (CCS)*, 2018.

[15] S. M. Werner, D. Perez, L. Gudgeon, A. Klages-Mundt, D. Harz, and W. J. Knottenbelt. SoK: Decentralized Finance (DeFi). In *ACM Conference on Advances in Financial Technologies (AFT)*, 2022.

[16] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *Int. Conference on Learning Representations (ICLR)*, 2023.

[17] L. Zhou, X. Xiong, J. Ernstberger, S. Chaliasos, Z. Wang, Y. Wang, K. Qin, R. Wattenhofer, D. Song, and A. Gervais. SoK: Decentralized Finance (DeFi) Attacks. In *IEEE Symposium on Security and Privacy (S&P)*, 2023.