International Journal of Research and Development in Engineering Sciences
ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/
Regular Issue, Vol. 8, Issue. 6 (November – December) , 2026 , Pages: 31 – 48
Received: 01 March 2026 ; *Accepted:* 30 October 2026 ; *Published:* 14 November 2026
https://doi.org/10.63328/IJRDES-V8RI6P4
https://www.youtube.com/watch?v=R7PHP4fFQic
https://archive.org/details/jsp-ijrdes-v8ri6p04
Original Paper : Applied Research – Experimental Research Paper

Trustworthy Autonomous Agents Through Evidence-Based Delegation: Formal Verification, Prototype Implementation, and Simulation Analysis

Marpu Jagadeesh

Department of Information Technology , St Francis College Brookyln , New York, USA

** Corresponding Author: Marpu Jagadeesh ;* jmarpu@sfc.edu

Abstract: Agents that call tools and act on their own are being deployed with very little supervision, and most of them are trusted once, at launch, and never looked at again. We think that is the wrong model. This paper describes Evidence-Based Delegation (EBD), in which an agent gets only as much authority as the checks on it can justify, and loses it quickly when a check fails. The design has four verification layers (identity, specification, execution and outcome) and a trust budget for each class of task. The budget rises slowly while checks come back clean and falls sharply when a violation is caught. We prove a bound on what an adaptive attacker can take from actions that run without review, work out how long promotion takes, and derive how much error an honest agent can show before it stops earning autonomy. We then tested the idea in a Monte-Carlo simulation (1,000 paired runs per setting, four agent behaviours, nine policies) and with a small prototype of the gateway and its tamper-evident log. In the simulation, EBD matched the loss of reviewing every action (113.4 against 114.3 loss units) with 51% fewer reviews, and it cut loss by 75 to 97% against static-permission and audit-only baselines. It also has clear weaknesses. It escalates a lot for unreliable agents, it does worse when risk is underestimated, and our proven bound is loose by two to three orders of magnitude. All results come from simulation under assumptions we state, so they describe how the mechanism behaves and say nothing certain about real deployed agents.

Keywords: autonomous agents, trust calibration, runtime verification, delegation, tamper-evident logging

1. Introduction

Suppose you let an agent clear your inbox, pay routine invoices and book your travel. In the first week you read everything it does. In the second you skim. By the third you have stopped looking. At no point did you decide to trust the agent. It just happened, and nobody checked whether the agent had earned it. This paper is about making that decision explicit and tying it to evidence. Agents are harder to trust than ordinary software for three reasons. The same input can lead to different action sequences, so a test suite says little about what will happen next week. An agent with tools can reach states its designers never listed. And what an agent says it did may be wrong, or deliberately misleading. The usual controls are testing before deployment, fixed permission scopes, and human approval for some actions. They help, but each one has a familiar weak spot. Testing cannot cover distribution shift. A fixed scope is either too tight to be useful or too loose to be safe. Human approval does not scale, and people get tired of approving [11]. Our position is that trust should be computed from verification evidence. It should apply to one class of tasks, be capped by what an error could cost, and be easy to withdraw. We call the resulting design Evidence-Based Delegation.

Section 4 presents the four verification layers, an evidence model in which observations fade with time, the trust budget that turns evidence into permissions, and a reference architecture with flowcharts. Section 5 proves a bound on what an adaptive adversary can extract and gives laws for promotion time, tolerable error and recovery. Sections 6 and 7 report a simulation study in which every policy sees the same random draws, with paired tests, ablations, sensitivity grids, and experiments on risk misestimation and reviewer fatigue. Section 8 describes a working prototype of the gateway and a hash-

Jack Sparrow Publishers © 2026, IJRDES, All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6, 2026 , https://doi.org/10.63328/IJRDES-V8RI1P4

chained log, including one tamper test that went against us. One note on scope. The experiments are simulations with simplified agents and assumed parameters (Appendix A). We did not test a real language-model agent, so the results show how the mechanism behaves under those assumptions and nothing more. We report the unfavourable findings alongside the favourable ones.

2. Background and Related Work

The human-factors literature on automation separates how reliable a system is from how much people trust it, and names the gap between them as the problem. The aim is calibration: trust that follows capability, so that people neither over-rely on a weak system nor ignore a good one [7, 11]. Organisational theory splits trustworthiness into ability, benevolence and integrity [9]. We adopt the calibration view and make it measurable, since our controller ties authority to evidence and Section 7.8 tests calibration directly.

On the safety side, reward hacking, unsafe exploration and distribution shift all apply to any agent that acts on its own [1]. Agents built on language models that mix reasoning with tool use [15, 12] are more capable and also easier to attack. Indirect prompt injection through retrieved content is one documented way to redirect them [4]. Fine-tuning on human preferences [10] shapes behaviour but gives no guarantee at run time. The case for keeping people able to intervene [5] is one reason we keep an escalation path rather than removing humans altogether.

The closest technical relatives are runtime verification and enforcement. Runtime verification watches executions and checks them against properties when the component itself is too opaque to verify in advance [8], and enforcement goes a step further by blocking violations, within limits that are well characterised [13]. Our gateway is an enforcement monitor around an agent we treat as a black box. For the log we borrow from distributed systems, which have long dealt with parties that fail in arbitrary ways [6], and from message authentication [2]. Proof systems [3] point toward checking claims without seeing everything, though we do not use them here.

What we did not find was a treatment that connects verification evidence to graduated, revocable authority along the whole path from identity to outcome, or one that measures the trade-off between safety and human effort against adaptive adversaries and tired reviewers. This paper tries to supply that, and says where it falls short.

3. Problem Statement and Existing System Threat Model

3.1. Setting and Notation

A principal (a person or organisation) gives tasks to an agent that acts through tools. Time is discrete. At step t the agent proposes an action a_t that belongs to a task class $c_t \in \{0, \dots, K-1\}$. If an action turns out to be a violation it has a true cost $\ell(a) \geq 0$. The controller does not know that cost, so it works with an estimate $\hat{r}(a)$, which we call the risk cost. A verifier produces evidence, and an autonomy controller turns evidence into a budget $B_c(t)$ for each class and into an autonomy level. Table 1 lists the notation.

Table. 1 Notation

Symbol	Meaning	Symbol	Meaning
K	Number Of Task Classes	π_c	Frequency Of Class c
$a_t,\ c_t$	Action And Its Class At Step t	$T,\ T_c$	Horizon; Number Of Class-c Actions
$\ell(a),\ \hat{r}(a)$	True Cost; Estimated Risk Cost	κ	Misestimation Factor, $\ell \leq \kappa\hat{r}$
$B_c(t),\ \phi_c(t)$	Trust Budget; $\phi_c = \log B_c$	$B^{\min},\ B_c^{\max}$	Budget Floor; Hard Cap
α	Reward Gain Per Unit Evidence	β	Penalty Per Detected Violation
γ	Staleness Decay Per Idle Step	$w_t \in [0,1]$	Evidence Strength At Step t
δ_c	Detection Probability Of A Violation	s_c	Outcome-Verification Sampling Rate
f	Contract Coverage (Share Of Violations	$d_g,\ d_o$	Gateway / Outcome Detection Power
	Detectable By Contract)	$L_c(t)$	Autonomy Level

3.2. Definitions

Definition 1 (Trust). The principal's willingness to rely on the agent for a class of tasks, under stated constraints, without direct supervision.

Definition 2 (Evidence). A recorded observation or proof $e = (\text{claim,source,layer,strength,time,class})$ that bears on a claim about the agent. Strength ranks evidence by how hard it is to fake: self-reported, then sampled, then independently recomputed, then proven.

An action is *autonomous* if it runs without per-action human approval because $\hat{r}(a) \leq B_c(t)$ and the gateway lets it through.

3.3. Threat model

We consider seven threats: impersonation of an agent or principal; specification gaming, where

Jack Sparrow Publishers © 2026, IJRDES, All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6, 2026 , https://doi.org/10.63328/IJRDES-V8RI1P4

the literal goal is met but the intent is violated; scope creep beyond the delegated authority; misreporting of what was done; prompt or data injection that redirects the agent; tampering with the log; and slow drift in behaviour that slips under fixed thresholds. The strongest adversary we simulate is an adaptive sleeper. It behaves honestly until a set time, then attacks exactly when the controller would let it act without review. When a tired reviewer is likely to miss an attack it also attacks reviewed actions, with lower probability. We assume nothing about the agent's internal reasoning or weights being visible. We do assume that the verifier code, the keys and the cryptographic primitives are trustworthy, and Section 9 discusses what happens if they are not.

3.4. Goals

We want four things. Autonomous actions should have a limited blast radius (G1, bounded harm). Every action should be attributable and reconstruct able (G2, accountability). Autonomy should follow demonstrated reliability (G3, calibration). And the cost of verification, in compute and human attention, should not wipe out the benefit of delegating (G4, usability).

4. The Evidence-Based Delegation Framework

4.1. Principle

An agent's authority at time t should depend on the verifiable evidence available at time t, apply to one class of tasks, and be capped by the cost of error.

In place of "this agent is approved" we get trust that is earned, scoped and revocable.

4.2. Four verification layers

The *identity* layer (L1) answers who is acting. Each agent instance holds a cryptographic identity tied to a model version, a configuration and a principal, and its actions are authenticated. The *specification* layer (L2) answers what the agent was asked to do. Tasks are written as delegation contracts that list the goal, allowed tools, resource limits, prohibited actions and success criteria. The *execution* layer (L3) answers what the agent actually did. A gateway mediates every tool call, checks it against the contract before it runs, and appends it to a tamper-evident log. The *outcome* layer (L4) answers whether the result is right, using tests, recomputation, a separate reviewer or sampled human review. The layers cover for each other. Identity without a specification gives attributable but unbounded behaviour. Execution monitoring without outcome checks can confirm that the rules were followed while the result is still wrong.

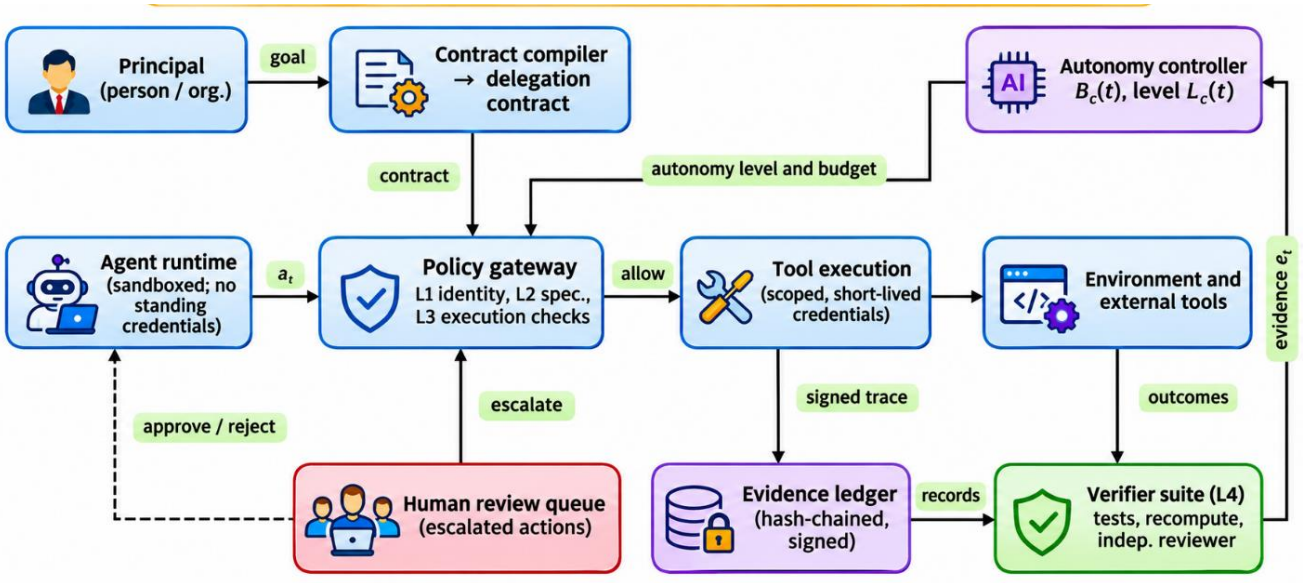

Figure. 1 Reference architecture. The agent never holds standing credentials, and every action goes through the policy gateway, which consults the autonomy controller. Executed actions leave a signed trace in the ledger. The verifier suite produces outcome evidence that feeds back to the controller. The dashed line is the human review path.

4.3. Reference architecture

Figure 1 shows the pieces. The agent runs in a sandbox with no direct access to tools or standing credentials. The policy gateway checks identity and contract, applies the controller's decision, and runs permitted calls with scoped, short-lived credentials. The evidence ledger is a hash chain that is anchored outside the system from time to time. The verifier suite implements L4, the autonomy controller keeps the budgets, and a contract compiler turns a principal's goals into machine-checkable contracts, asking for clarification when a goal is ambiguous.

Jack Sparrow Publishers © 2026, IJRDES, All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6, 2026 , https://doi.org/10.63328/IJRDES-V8RI1P4

4.4. Deciding on one action

Algorithm 1 and Figure 2 show how the gateway handles a single action. An action with a bad signature or outside the contract is denied outright. Otherwise the gateway estimates its risk cost. If that cost fits within the current budget, and the evidence for the class is fresh, the action runs on its own. If not, it goes to a human reviewer, and runs only if the reviewer approves.

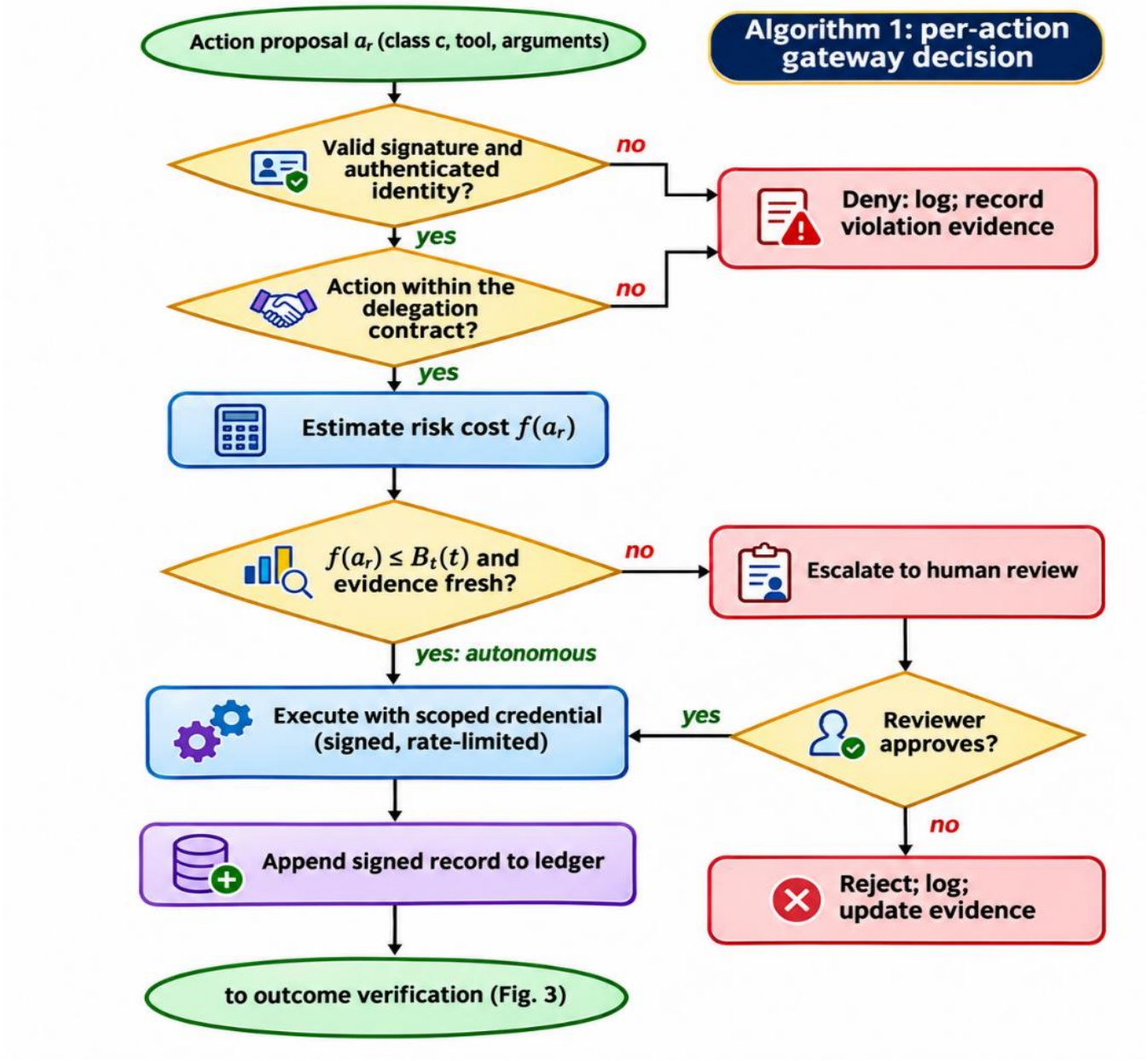

Figure. 2 Flowchart of the per-action gateway decision (Algorithm 1).

4.5. Evidence and the trust budget

Evidence comes from four places, each with a strength: a clean pass through the gateway (w_g), a sampled outcome check (w_o), a human-reviewed approval (w_h) and, where available, independent certification.

When several sources apply to one step, we take the strongest. For each task class the controller keeps a log-budget $\phi_c(t) = \log B_c(t)$. After a class-c action at step t it is updated as

$$\phi_c(t+1) = \mathrm{clip}\big(\phi_c(t) + \alpha\, w_t\, \mathbb{1}[\text{no detection}] - \beta\, \mathbb{1}[\text{violation detected}],\ \log B_c^{\min},\ \log B_c^{\max}\big), \quad (1)$$

and every class that got no evidence at that step loses γ to staleness.

Three choices in this rule matter. Working on the log scale makes the update behave the same way for classes whose risk costs differ by orders of magnitude.

An undetected violation looks like a clean action to the controller, so everything below depends on how good the detectors are. And we require $\beta \gg \alpha w$, so that trust is slow to earn and quick to lose.

A hard cap $B_c^{\max}$ stops budgets from growing without limit. For the riskiest class we set the cap below the median risk cost, so most of those actions always need a human.

Jack Sparrow Publishers © 2026, IJRDES, All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6, 2026 , https://doi.org/10.63328/IJRDES-V8RI1P4

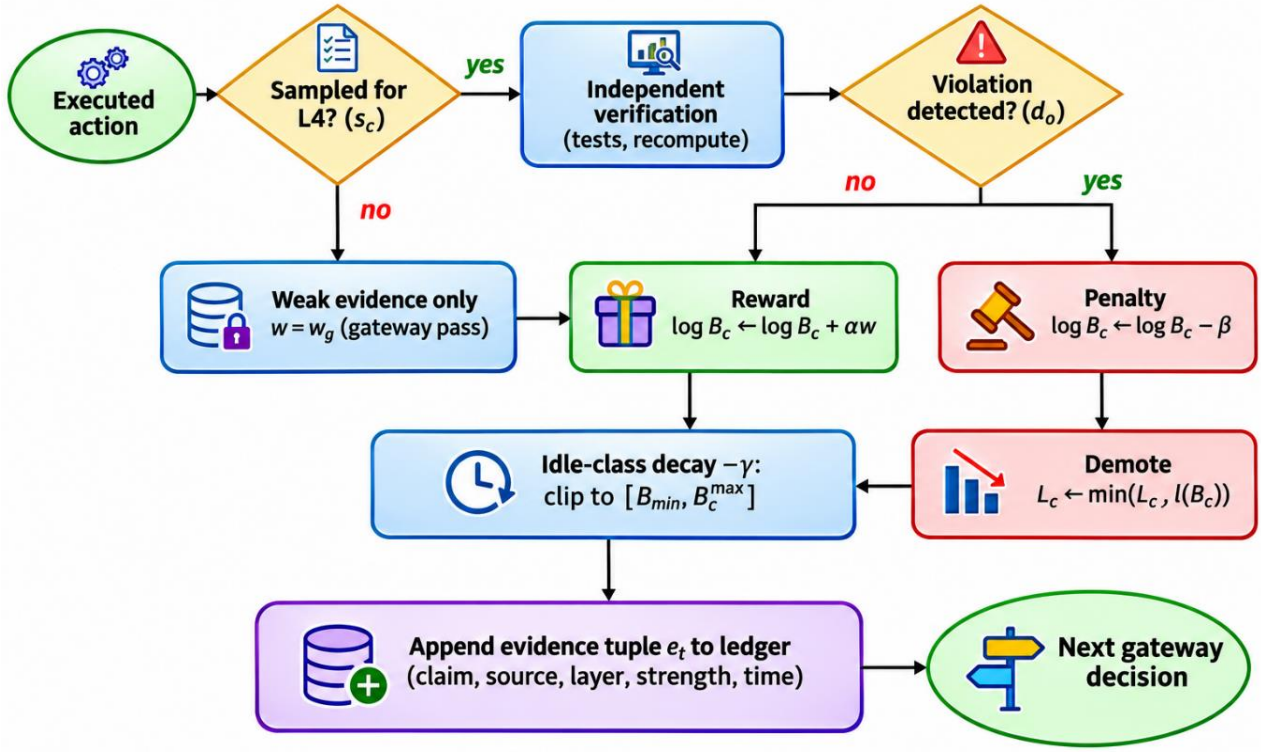

Figure. 3 Flowchart of evidence collection and budget update (Algorithm 2, Eq. (1)). Penalties from outcome checks arrive after a detection lag.

4.6. Autonomy levels

The budget maps to five levels (Figure 4). L0 only suggests. L1 executes with approval for each action. L2 lets low-risk actions run freely and sends the rest for approval. L3 gives broad autonomy within the contract, with periodic audit. L4 allows autonomy with review after the fact. Moving up needs a budget above a threshold θ_k and evidence that is diverse, meaning successes on varied inputs and not repeats of one. Moving down happens at once after a verified violation. In our experiments the test $\hat{r}(a) \leq B_c(t)$ does the job of the level map, and the discrete levels are there to make the system easier for operators to read.

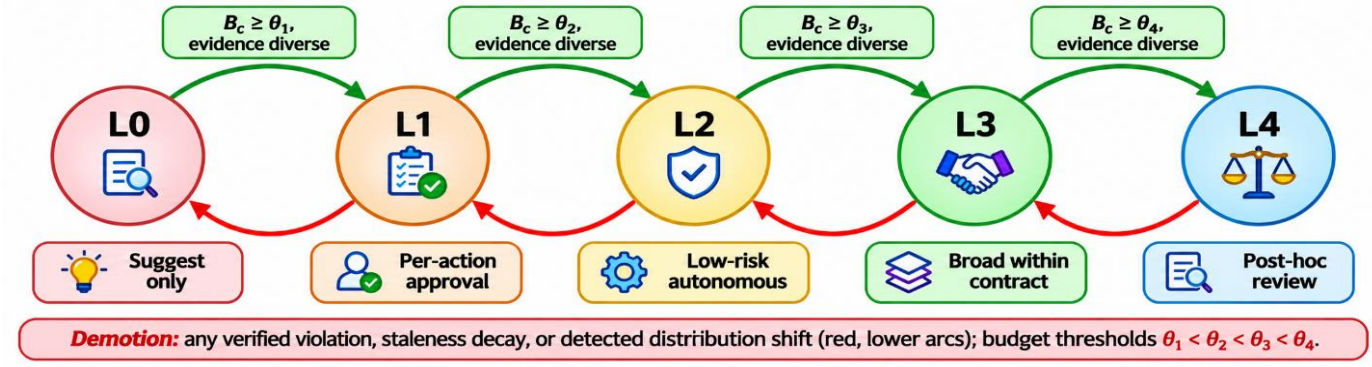

Figure. 4 Autonomy levels as thresholds on the log trust budget. Green arcs are promotions on sufficient, diverse evidence. Red arcs are demotions after verified violations, staleness or detected distribution shift.

4.7. Properties we test

We check five properties. P1: the loss from autonomous actions is bounded in terms of the budget (Proposition 1 and Theorem 1, tested in Section 7.7). P2: every executed action can be traced to an authenticated agent, and P3: tampering with the log is detectable. Both are tested on the prototype (Section 8). P4: a detected violation never raises autonomy, which holds by construction in Eq. (1). P5: granted autonomy is negatively related to measured error (Section 7.8).

5. Theoretical Analysis

A violation is an action that breaches the intent of the contract. We fix a class c and drop the subscript where it is clear. We make four assumptions.

Jack Sparrow Publishers © 2026, IJRDES, All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6, 2026 , https://doi.org/10.63328/IJRDES-V8RI1P4

A1 (mediation). An action runs autonomously only if $\hat{r}(a) \le B_c(t)$.

A2 (bounded misestimation). $\ell(a) \le \kappa\, \hat{r}(a)$ for some constant $\kappa \ge 1$.

A3 (detection). Each violating action attempted autonomously is detected with probability at least δ_c, independently of the past, and a detection applies the penalty β at once.

A4 (dynamics). The budget follows Eq. (1) with $w_t \le \bar{w} \le 1$.

Lemma 1 (Combining detection layers). If layers detect a violation through independent mechanisms with probabilities $\delta^{(1)}, \dots, \delta^{(m)}$, the overall detection probability is $\delta = 1 - \prod_{i=1}^{m}\left(1 - \delta^{(i)}\right)$. For the gateway and outcome layers, $\delta_c = 1 - \left(1 - f\, d_g\right)\left(1 - s_c\, d_o\right)$.

Proof. A violation gets through only if every layer misses it, and independence turns that into a product of miss probabilities. The gateway catches a violation with probability $f\, d_g$: the violation has to be detectable by contract (share f) and the check has to fire. The outcome verifier catches it with probability $s_c\, d_o$: it has to be sampled and then detected. ■

Proposition 1 (Single-action bound). Under A1 and A2, every autonomously executed action satisfies $\ell(a) \le \kappa\, B_c(t) \le \kappa\, B_c^{\max}$.

Proof. A1 and A2 give $\ell(a) \le \kappa \hat{r}(a) \le \kappa B_c(t)$, and the clip in Eq. (1) gives $B_c(t) \le B_c^{\max}$.

This bounds one action, but an adversary can repeat violations. The next result bounds the total loss against any adaptive strategy.

Theorem 1 (Exploitation bound). Under A1 to A4, with T_c class- c actions, $\Lambda = \log(B_0/B^{\min})$ and any (possibly adaptive) agent strategy, the expected loss from autonomously executed class-c actions satisfies

$$\mathbb{E}[\text{loss}_c] \;\le\; \kappa\left[B_c^{\max}\,\frac{\alpha\bar{w}\,T_c + \Lambda}{\delta_c\,\beta} \;+\; B^{\min} e^{\beta}\,T_c\right]. \qquad (2)$$

Proof. Write $\phi_t = \log B_c(t)$ and let N be the number of violations attempted autonomously. Call an attempt at time t a *floor event* if $\phi_t < \log B^{\min} + \beta$, and let N' and D' count the non-floor attempts and the detected non-floor attempts. A detected non-floor attempt takes the full penalty β, because nothing is clipped at the floor. Gains come only from clean evidence steps, of which there are at most T_c, each worth at most $\alpha\bar{w}$. Clipping at the cap, staleness decay and floor-event penalties can only push ϕ lower than this count assumes. So

$$\phi_T \;\le\; \phi_0 + \alpha\bar{w}\,T_c - \beta D'.$$

Since $\phi_T \ge \log B^{\min}$ and $\phi_0 = \log B_0$, this rearranges to $\beta D' \le \alpha\bar{w}\,T_c + \Lambda$. By A3 each non-floor attempt is detected with conditional probability at least δ_c given the history, so $D' - \delta_c N'$ is a submartingale with non-negative expected increments. That gives $\delta_c\,\mathbb{E}[N'] \le \mathbb{E}[D'] \le (\alpha\bar{w}\,T_c + \Lambda)/\beta$. A non-floor attempt has $\hat{r} \le B_c \le B_c^{\max}$, so by A2 its loss is at most $\kappa B_c^{\max}$. A floor attempt has $\hat{r} \le B_c(t) < B^{\min} e^{\beta}$, so its loss is at most $\kappa B^{\min} e^{\beta}$, and there are at most T_c of them. Adding the two parts gives Eq. (2). ■

Corollary 1 (Dependence on the penalty ratio). Let $\rho = \beta/(\alpha\bar{w})$. Apart from the start-up term Λ, the loss per action is at most $\kappa\left[B_c^{\max}/(\delta_c\rho) + B^{\min} e^{\beta}\right]$. For fixed δ_c the first term falls as $1/\rho$.

Remark 1 (Delayed detection). Suppose outcome detections take effect after a delay of at most Δ steps. Then at most Δ further attempts can happen before the last pending penalties land, which adds at most $\kappa B_c^{\max}\Delta$ to Eq. (2). We have not proved this extension. Section 7.7 checks it empirically. The bound covers only autonomous losses. Escalated actions rely on the reviewer, whose miss rate sits outside the budget mechanism (see the fatigue results in Section 7.9).

Proposition 2 (Promotion time). Suppose class c is acted on at rate π_c per step, with average evidence strength $\bar{w}_c$ on its clean steps and decay γ on idle steps. In the fluid approximation, the number of steps needed to raise the budget from B_0 to B^* is

$$\tau_c(B^*) \;\approx\; \frac{\log(B^*/B_0)}{\pi_c\,\alpha\,\bar{w}_c - \gamma(1-\pi_c)} \qquad \text{if the denominator is positive,} \qquad (3)$$

and B^* is unreachable in expectation otherwise. A refined model replaces $\bar{w}_c$ by $w_{\text{eff}}(\phi) = P_{\text{esc}}(\phi)\, w_h\left(1 - h_{\text{fp}}\right) + P_{\text{aut}}(\phi)\left[s_c w_o + (1 - s_c) w_g\right]$, with $P_{\text{aut}}(\phi) = \Phi\left((\phi - \log m_c)/\sigma_{\text{tot}}\right)$. It also subtracts the expected penalty drift $\beta\, p_c\,[P_{\text{aut}}\delta_c + P_{\text{esc}} h]$ and integrates $\mathrm{d}\phi/g(\phi)$ numerically. Here m_c is the median risk of the class, $\sigma_{\text{tot}}^2 = \sigma^2 + \sigma_\epsilon^2$ combines the spread of risk and the noise in the estimate, and Φ is the normal cdf.

Derivation. Class c gets evidence with probability π_c per step and idles otherwise, so the expected change in ϕ_c per step is $\pi_c\alpha\bar{w}_c - \gamma(1-\pi_c)$. Dividing the required rise $\log(B^*/B_0)$ by this drift gives Eq. (3). In the refined model the drift per class- c action is $g(\phi) = \alpha w_{\text{eff}}(\phi) - \text{penalty}(\phi) - \gamma(1-\pi_c)/\pi_c$. The number of class-$c$ actions is $\int \mathrm{d}\phi/g(\phi)$, and dividing by π_c converts that to steps.

Proposition 3 (Tolerable violation rate and recovery time). Let ν_c be the probability that an autonomous class-c action is a detected violation. The log-budget has positive expected drift (ignoring decay and clipping) if and only if $\nu_c < \nu^* = \alpha\bar{w}/\beta = 1/\rho$. After one detected violation, the expected number of class-c steps needed to win back the lost budget is ρ/π_c.

Proof. The expected change per action is $\alpha\bar{w}(1-\nu_c) - \beta\nu_c$. It is positive iff $\nu_c < \alpha\bar{w}/(\beta + \alpha\bar{w})$, which is approximately $\alpha\bar{w}/\beta$ when $\beta \gg \alpha\bar{w}$. A penalty of β is repaid at $\alpha\bar{w}$ per clean evidence step, so it takes $\beta/(\alpha\bar{w}) = \rho$ class-c actions, or ρ/π_c steps.

Jack Sparrow Publishers © 2026, IJRDES, All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6, 2026 , https://doi.org/10.63328/IJRDES-V8RI1P4

What the theory says in practice. A larger ρ shrinks the loss an attacker can extract (Corollary 1). It also lowers the violation rate an honest agent can have before it stops earning autonomy (Proposition 3). Rarely used, high-risk classes promote slowly (Proposition 2). So choosing ρ means trading safety against autonomy, and Section 7.6 looks at that trade-off in the simulation.

6. Experimental Setup and Implementation

6.1. Simulator

We wrote a vectorised Monte-Carlo simulator of the framework. Each run has $T = 2{,}000$ actions in $K = 4$ classes with frequencies $\pi = (0.45, 0.30, 0.18, 0.07)$. The median risk costs are $(1, 5, 20, 100)$ loss units, log-normal with dispersion $\sigma = 0.5$.

Class 0 stands for reads and summaries, and class 3 for high-impact operations that are mostly irreversible. The controller sees a noisy estimate $\hat{r} = r\,e^{\epsilon}/b$, with $\epsilon \sim \mathcal{N}(0, 0.2^2)$ and bias b (default $b = 1$). Some violations can be detected from the contract (share $f = 0.6$) and the rest are semantic.

A violating action that runs autonomously costs r, unless an outcome check catches it later. In that case a fraction $\mu_c = (0.9, 0.7, 0.4, 0.1)$ of the loss is recovered, less the more time has passed.

Outcome detections arrive after an exponential lag with mean 15 actions, cut off at 200. Reviewers catch a violation with probability $h = 0.9$ and wrongly reject a clean action with probability 0.02.

One modelling choice matters a great deal: undetected violations look clean to the controller, so the budget can be fooled.

6.2. Agent behaviours

The *reliable* agent is honest, with base violation probabilities of $(0.2, 0.4, 0.8, 1.5)\%$ by class, and it meets injected content on 2% of actions, to which it gives in with probability 0.05. The *drifting* agent is reliable until step 600 and then its violation probabilities ramp up to $12\times$ over the next 1,100 steps. The *adaptive sleeper* is nearly honest (0.05%) until step 500.

After that it attacks, with probability 0.5 per eligible action in classes 1 to 3, only when the controller would let it run autonomously. On escalated actions it attacks with probability $0.5(1 - h_{\text{eff}})$.

The *injection-prone* agent meets injected content on 8% of actions and gives in with probability 0.5. The "fleet" figures average the four behaviours with equal weight.

6.3. Policies

Always-approve sends every action to a human. **Static-narrow** and **static-broad** use fixed allow-list thresholds ($\tau = 10$ and $\tau = 60$ risk units) with no monitoring, and escalate anything above the threshold. **Post-hoc audit** lets everything run and audits 10% of actions later.

EBD (full) uses the gateway (L3), outcome verification (L4) and the adaptive budget, with $\alpha = 0.03$, $\beta = 1.5$ (so $\beta/\alpha = 50$), $\gamma = 3 \times 10^{-4}$, $B_0 = 2$, $B^{\min} = 0.5$, caps $(30, 60, 120, 80)$ and L4 sampling rates $(0.2, 0.4, 0.7, 0.9)$. We also ran four ablations: EBD without L3, EBD without L4, EBD with a fixed budget (monitors on, no adaptation), and EBD with $\beta = \alpha$ (symmetric response).

6.4. Metrics and statistics

For each run we record total loss, the number of harmful actions that executed, the largest single loss, the escalation rate (reviews per action), task success (clean and executed), modelled latency and verification compute. Every setting uses $1{,}000$ independent seeds.

All policies see identical task streams and random draws, so comparisons are paired. We report means with 95% normal confidence intervals. For policy comparisons we also give the paired difference, the percentage of seeds in which EBD has the lower loss, and a Wilcoxon signed-rank p-value [14]. Calibration is measured with Spearman's ρ. Parameter values are in Appendix A.

7. Results and Analysis

7.1. Main comparison

Table 2 and Figure 5 give the fleet-level results. EBD (full) has a mean loss of 113.4, which is practically the same as always-approve at 114.3.

The paired difference is -0.8%, and EBD has the lower loss in only 55% of seeds. With $n = 1{,}000$ the difference is detectable (Wilcoxon $p < 0.001$) but small. EBD gets there while escalating 49% of actions instead of 100%, which is 51% fewer reviews. Against the static and audit-only baselines, EBD lowers loss by 75% (static-narrow, 459.3), 93% (static-broad, 1,561) and 97% (post-hoc audit, 3,429).

Two things temper this. EBD is not the most frugal option: static-narrow escalates only 26% of actions against EBD's 49%, and which system is cheaper overall depends on what a human review costs (Section 7.11). EBD also lets more harmful actions through than always-approve (4.3 against 2.4 per 1,000) while ending up with a similar total loss.

The violations that slip past EBD are capped by the budget and so tend to be small, whereas a human reviewer also stops the small ones.

Jack Sparrow Publishers © 2026, IJRDES, All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6, 2026 , https://doi.org/10.63328/IJRDES-V8RI1P4

Table. 2 Fleet-level results (mean over four agent behaviours; 1,000 paired seeds; ± is the 95% CI half-width). Loss is in risk units per run of 2,000 actions. "Harmful" counts violations that executed per 1,000 actions. "Max loss" is the mean largest single loss. "Success" is the share of actions that were clean and executed. Latency is modelled added seconds per action (reviewer wait 30 s, gateway 40 ms).

Policy	Loss	Harmful/1k	Max loss	Escalation	Success	Latency (s)
Always-approve	114 ± 4	2.4 ± 0.0	66.4	1.000	0.956	30.00
Static-narrow	459 ± 4	39.7 ± 0.1	67.7	0.263	0.946	7.88
Static-broad	1,561 ± 8	63.0 ± 0.2	87.2	0.062	0.933	1.86
Post-hoc audit	3,429 ± 20	70.9 ± 0.2	221.5	0.000	0.929	0.00
EBD (full)	113 ± 4	4.3 ± 0.1	62.8	0.489	0.966	14.70
EBD −L3	148 ± 4	9.9 ± 0.1	64.5	0.428	0.966	12.85
EBD −L4	126 ± 4	4.2 ± 0.1	67.8	0.510	0.965	15.34
EBD fixed budget	529 ± 5	25.9 ± 0.1	70.8	0.080	0.934	2.45
EBD $\beta = \alpha$	443 ± 5	24.0 ± 0.1	68.6	0.159	0.936	4.80

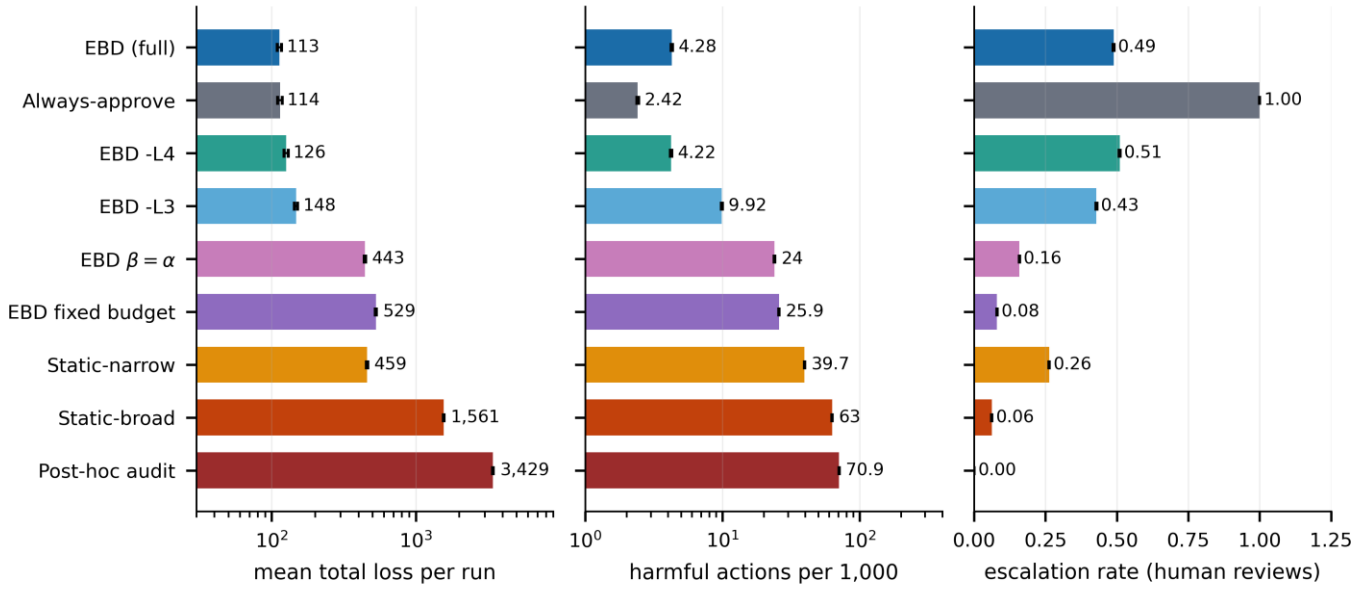

Figure. 5 Fleet-level comparison on three outcome measures (log scale for loss and harmful actions; bars show 95% CIs). EBD matches the loss of reviewing everything while escalating about half as often.

7.2. Results by behaviour

Table 3 and Figure 6 split the results by agent behaviour. With the reliable agent, always-approve has the lowest loss (32) and EBD is close (40) with about a quarter of the reviews (24%). Static-narrow is also competitive (49), since little can go wrong.

The picture changes once an agent misbehaves. Against the adaptive sleeper, EBD loses 107, compared with 1,231 for static-narrow, 4,730 for static-broad and 10,180 for post-hoc audit. Static-broad does about 44 times worse than EBD here, because static policies never revisit their grants. Against the drifting agent, EBD (175) stays level with always-approve (177) while static-narrow reaches 258.

The injection-prone agent shows a limit. EBD's loss (131) is again close to always-approve (141), but only because EBD escalates 79% of actions. This agent violates often enough that it rarely earns autonomy. EBD is doing the right thing, but it saves little effort when the agent is simply unreliable.

7.3. Paired comparison

Table 4 gives paired differences in fleet loss between EBD (full) and each alternative. The differences against the non-adaptive and audit-only policies are large and significant, and the ablations are also significantly worse than full EBD. Removing the gateway raises fleet loss by 30%, removing outcome verification by 11%, freezing the budget by 366%, and making the response symmetric ($\beta = \alpha$) by 291%.

Figure 7 shows the ablations for each behaviour. The components do different jobs. Removing the gateway (L3) hurts most for the reliable agent (55%) and the sleeper

Jack Sparrow Publishers © 2026, IJRDES, All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6, 2026 , https://doi.org/10.63328/IJRDES-V8RI1P4

(41%), where contract-detectable attacks would otherwise be blocked before they run and penalised at once. Removing outcome verification (L4) costs 8% for the reliable agent and a little more where violations are mostly semantic: 10% for drifting, 12% for injection-prone and 14% for the sleeper. Freezing the budget costs a moderate 34% for the reliable agent but is disastrous against the sleeper (1,503 against 107, a factor of 14). Monitors without adaptation still let an attacker work indefinitely. The symmetric response ($\beta = \alpha$, 1,212) fails in the same way. What contains the adaptive adversary is the asymmetry, not just the presence of penalties.

Table. 3 Mean total loss by agent behaviour (± 95% CI; 1,000 paired seeds).

Policy	Reliable	Drifting	Sleeper	Injection-prone
Always-approve	32 ± 4	177 ± 9	107 ± 6	141 ± 7
Static-narrow	49 ± 4	258 ± 9	1,231 ± 8	299 ± 8
Static-broad	132 ± 5	678 ± 12	4,730 ± 20	705 ± 10
Post-hoc audit	344 ± 12	1,794 ± 27	10,180 ± 56	1,398 ± 23
EBD (full)	40 ± 4	175 ± 8	107 ± 6	131 ± 7
EBD −L3	63 ± 4	223 ± 9	151 ± 6	153 ± 7
EBD −L4	44 ± 4	192 ± 9	122 ± 6	147 ± 7
EBD fixed budget	54 ± 4	283 ± 9	1,503 ± 11	275 ± 8
EBD $\beta = \alpha$	47 ± 4	275 ± 9	1,212 ± 11	238 ± 7

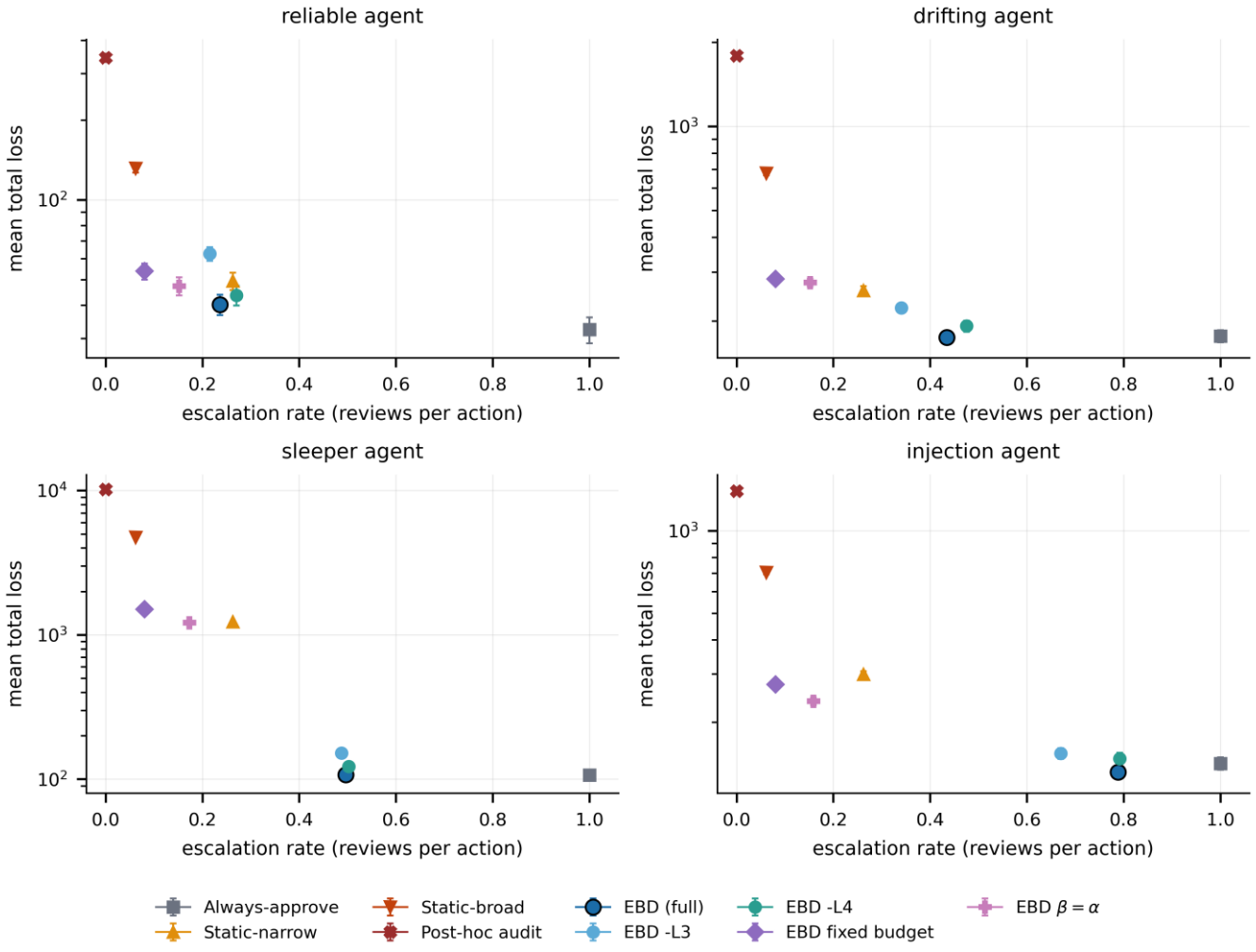

Figure. 6 Loss against escalation rate for each agent behaviour (lower left is better; error bars are 95% CIs). Static and audit-only policies sit at low oversight and high loss. Always-approve sits at full oversight. The adaptive EBD variants lie near the lower envelope.

Jack Sparrow Publishers © 2026, IJRDES, All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6, 2026 , https://doi.org/10.63328/IJRDES-V8RI1P4

Table. 4 Paired comparison of EBD (full) with each alternative on fleet loss. "Diff" is EBD minus the alternative (negative favours EBD). "Wins" is the percentage of seeds in which EBD has strictly lower loss. p is a two-sided Wilcoxon signed-rank test.

Alternative	Diff (units)	Relative	EBD wins	p
Always-approve	-1.0	-0.8%	54.9%	$< 10^{-3}$
Static-narrow	-345.9	-75.3%	100.0%	$< 10^{-3}$
Static-broad	-1,447.8	-92.7%	100.0%	$< 10^{-3}$
Post-hoc audit	-3,315.7	-96.7%	100.0%	$< 10^{-3}$
EBD −L3	-34.2	-23.2%	99.9%	$< 10^{-3}$
EBD −L4	-12.9	-10.2%	94.9%	$< 10^{-3}$
EBD fixed budget	-415.4	-78.6%	100.0%	$< 10^{-3}$
EBD $\beta = \alpha$	-329.9	-74.4%	100.0%	$< 10^{-3}$

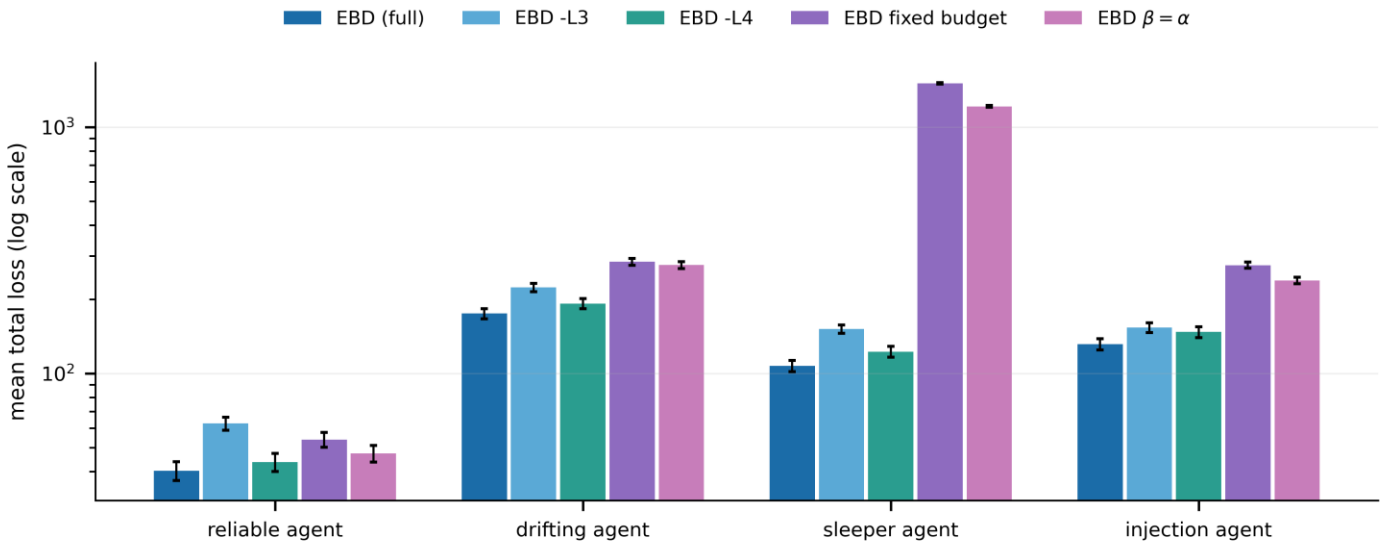

Figure. 7 Ablation of EBD components across agent behaviours (log scale, 95% CIs). Freezing the budget or making the response symmetric does serious damage when the agent changes behaviour.

7.4. Budget dynamics

Figure 8 shows geometric-mean budget trajectories with 10 to 90% bands. For the reliable agent, the budgets of classes 1 and 2 climb steadily and level off near their caps, while the highest-risk class 3 grows slowly and stays low, which is what we intended.

For the drifting agent the budgets rise until drift begins and then decay as detected violations pile up. For the sleeper they collapse within tens of steps of activation and stay down. Figure 9 shows the effect on cumulative loss. After activation, static, fixed-budget and audit-only policies run away from EBD, whose curve stays close to always-approve.

7.5. Sensitivity Analysis

Figure 10 shows fleet-level sensitivity grids with 200 seeds per cell. The penalty ratio matters most. Loss drops sharply as β/α goes from 1 (between 430 and 522, depending on detector power) to 10 (173), and flattens beyond about 30 (124; 116 at 200). Escalation climbs the whole way, from 0.16 to 0.55. This is the trade-off the theory predicts: a larger ratio buys safety by withdrawing autonomy more readily.

Outcome-detection strength d_o matters less than we expected. At a ratio of 60, loss is 127 at $d_o = 0.3$ and 116 at 0.95, because the gateway, the reviewer and the sampling rate already carry much of the detection load. Contract coverage f and the L4 sampling scale have moderate effects of similar size.

Going from the weakest corner ($f = 0.2$, no sampling; 170) to the strongest ($f = 0.95$, sampling × 1.5; 105) lowers loss by roughly 40%.

Jack Sparrow Publishers © 2026, IJRDES, All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6, 2026 , https://doi.org/10.63328/IJRDES-V8RI1P4

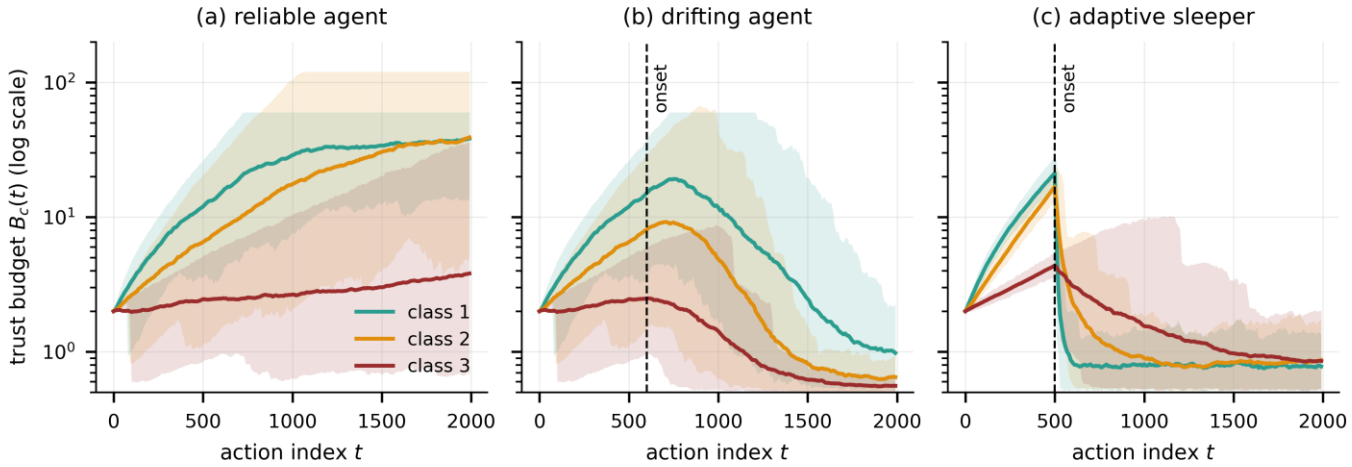

Figure. 8 Trust-budget trajectories under EBD (geometric mean with 10 to 90% band across 300 runs). Dashed lines mark the start of drift or attack.

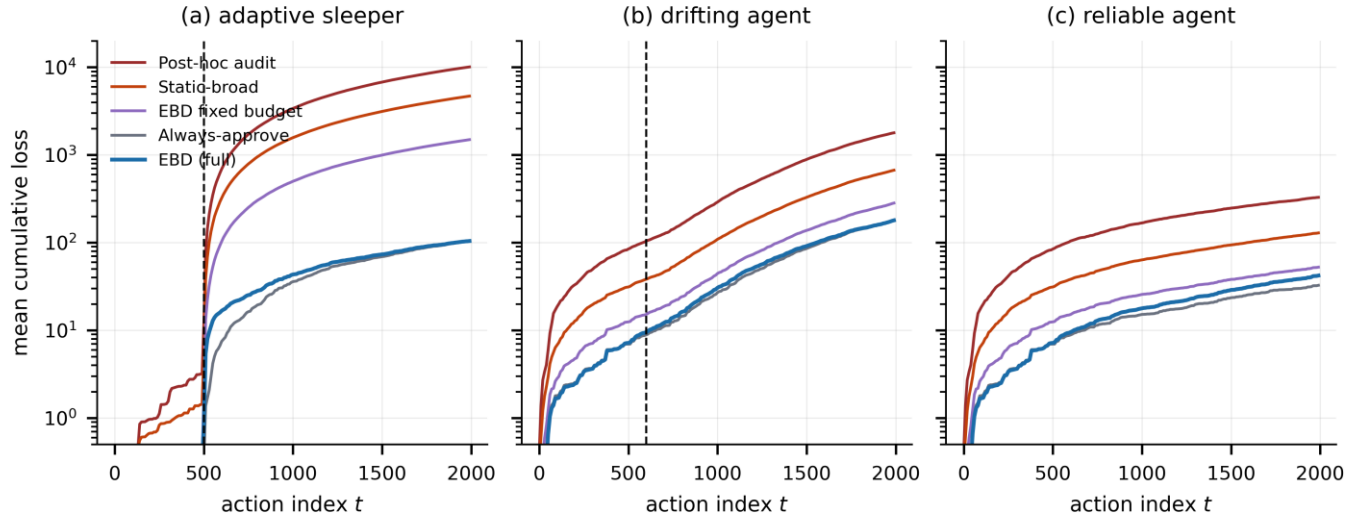

Figure. 9 Mean cumulative loss over time (log scale, 500 runs). After sleeper activation (a) or the start of drift (b), non-adaptive policies diverge while EBD stays near the always-approve curve.

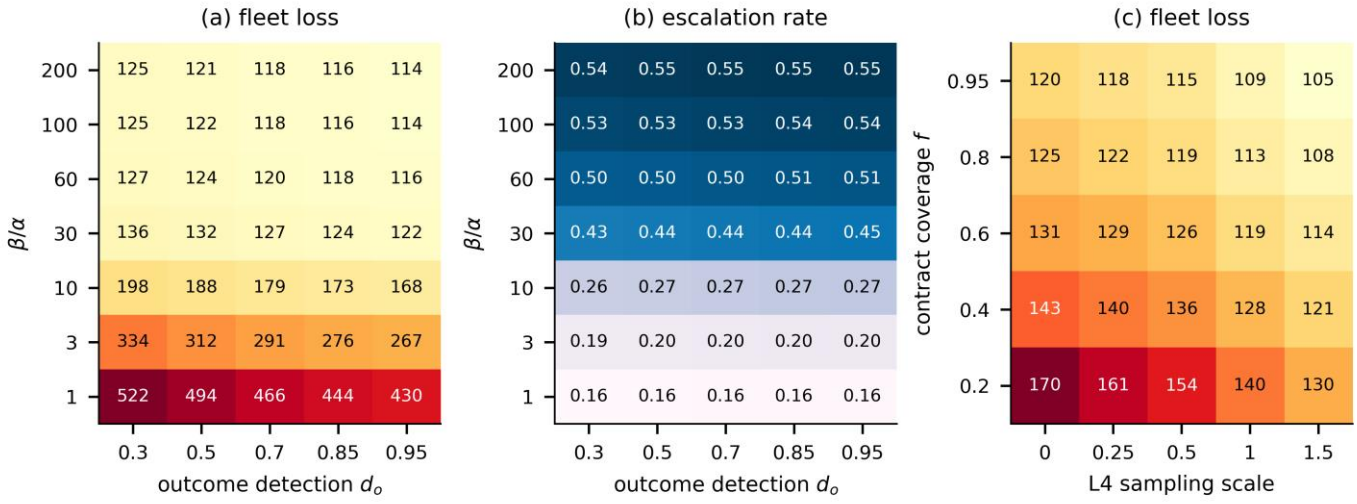

Figure. 10 Sensitivity of fleet loss and escalation to (a, b) the penalty ratio β/α and outcome-detection strength, and (c) contract coverage and L4 sampling scale. Cell values are means over four behaviours and 200 seeds.

Jack Sparrow Publishers © 2026, IJRDES, All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6, 2026 , https://doi.org/10.63328/IJRDES-V8RI1P4

7.6. Checking the theory against the simulation

Exploitation bound. Figure 11(a) compares the right-hand side of Eq. (2) with the simulated unsupervised loss of the adaptive sleeper at seven penalty values (Table 5). The bound is summed over classes, with $\delta_c = (0.65, 0.72, 0.83, 0.90)$ from Lemma 1 and $\bar{w} = 0.9$. It holds in all seven settings, including with outcome-detection delays, but it is loose by a factor of 118 to 1,345. The proof has to cover an adversary that attacks at the cap every time, and its floor term $B^{\min} e^{\beta} T_c$ grows with β. The shape is still informative. The simulated loss falls with a log-log slope of -1.26 over $\beta \in [0.03, 1]$, close to the $1/\beta$ scaling in Corollary 1. So the bound is useful for choosing the penalty ratio, not for predicting loss.

Table. 5 Exploitation bound (Theorem 1, summed over classes) against simulated unsupervised loss for the adaptive sleeper (300 seeds; $\alpha = 0.03$).

β	β/α	Bound	Simulated	Slack ×
0.03	1	157,692	1,136.7	139
0.1	3	48,104	407.4	118
0.3	10	17,016	86.3	197
0.6	20	9,655	27.1	356
1	33	7,418	16.3	456
1.5	50	7,615	12.8	595
2.5	83	14,062	10.5	1,345

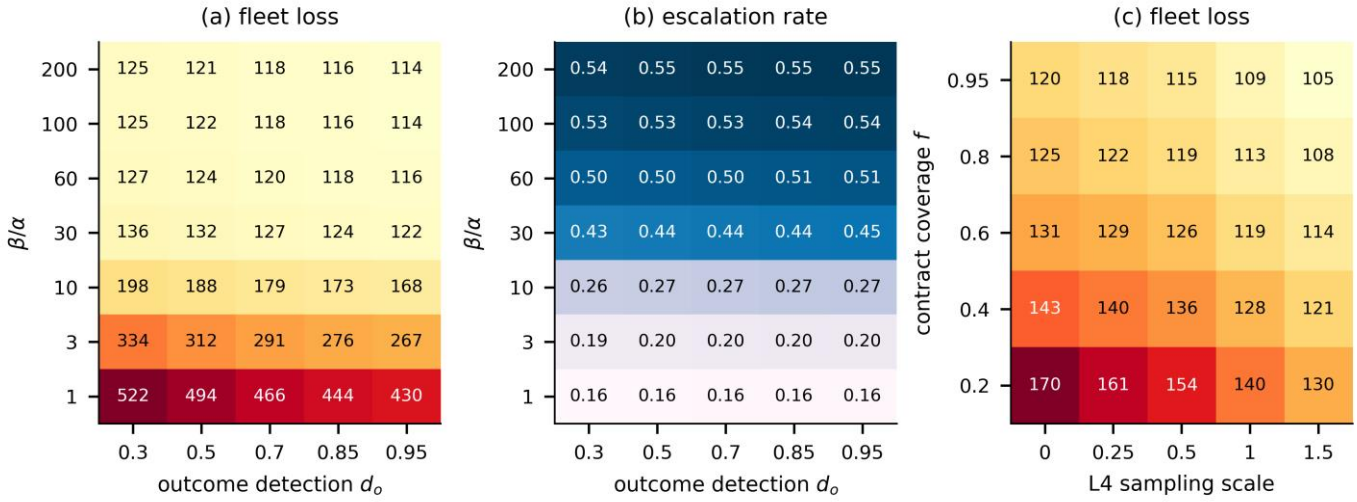

Figure. 10 Sensitivity of fleet loss and escalation to (a, b) the penalty ratio β/α and outcome-detection strength, and (c) contract coverage and L4 sampling scale. Cell values are means over four behaviours and 200 seeds.

7.7. Checking the theory against the simulation

Exploitation bound. Figure 11(a) compares the right-hand side of Eq. (2) with the simulated unsupervised loss of the adaptive sleeper at seven penalty values (Table 5). The bound is summed over classes, with $\delta_c = (0.65, 0.72, 0.83, 0.90)$ from Lemma 1 and $\bar{w} = 0.9$. It holds in all seven settings, including with outcome-detection delays, but it is loose by a factor of 118 to 1,345. The proof has to cover an adversary that attacks at the cap every time, and its floor term $B^{\min} e^{\beta} T_c$ grows with β. The shape is still informative. The simulated loss falls with a log-log slope of -1.26 over $\beta \in [0.03, 1]$, close to the $1/\beta$ scaling in Corollary 1. So the bound is useful for choosing the penalty ratio, not for predicting loss.

Promotion time. Table 6 and Figure 11(b) compare Proposition 2 with simulation for the reliable agent. The first-order law is too optimistic, because it assumes strong evidence on every action while evidence gets weaker as more actions run autonomously. The refined model is too pessimistic. The simulated median falls between the two for classes 1 and 2. For class 3 the refined model says the target is effectively out of reach, and only 8% of simulated runs get there within the horizon. The refined model overestimates the class 1 and 2 times by roughly 30%, so it gets the picture right but not the numbers.

Table. 5 Exploitation bound (Theorem 1, summed over classes) against simulated unsupervised loss for the adaptive sleeper (300 seeds; $\alpha = 0.03$).

β	β/α	Bound	Simulated	Slack ×
0.03	1	157,692	1,136.7	139
0.1	3	48,104	407.4	118
0.3	10	17,016	86.3	197
0.6	20	9,655	27.1	356
1	33	7,418	16.3	456
1.5	50	7,615	12.8	595
2.5	83	14,062	10.5	1,345

Jack Sparrow Publishers © 2026, IJRDES, All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6, 2026 , https://doi.org/10.63328/IJRDES-V8RI1P4

Table. 6 Steps to reach the target budget B^* for the reliable agent: fluid predictions against simulation (500 seeds). "Reached" is the share of runs that reach B^* within 2,000 steps.

Class	B^*	First-order	Refined	Sim. median	Reached
1	5	119	188	142	100%
2	20	511	1,023	820	93%
3	40	1,909	33,732	$> 2{,}000$	8%

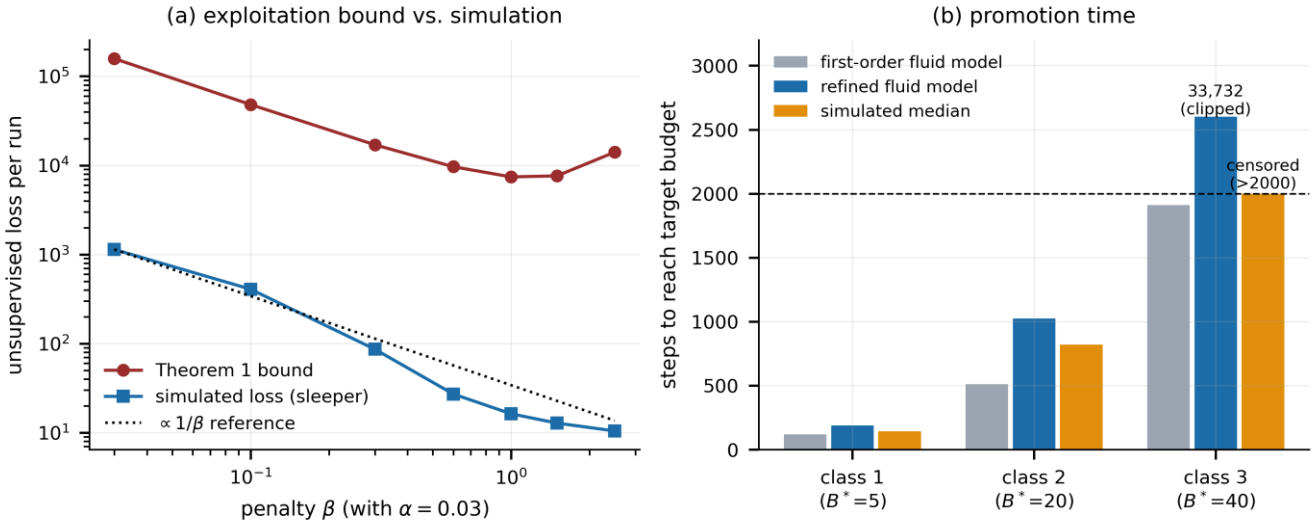

Figure. 11 Theory against simulation. (a) The exploitation bound (Theorem 1) holds but is conservative, and simulated loss follows roughly $1/\beta$. (b) The first-order and refined fluid models bracket the simulated median promotion time. Class 3 is effectively unreachable, in line with simulation (censored at T).

Tolerable violation rate. Table 7 applies Proposition 3 to the reliable agent. With $\beta/\alpha = 50$ and $\bar{w} = 0.88$, the tolerable rate of detected violations is $\nu^* = 1.76\%$ per autonomous action, and repaying one penalty takes ρ/π_c steps: 189 for class 1 and 812 for class 3. The reliable agent's effective detected-violation rate is below ν^* in every class, but for class 3 the margin is thin (a drift of +0.0052 per action). That is why class 3 grows so slowly.

7.8. Calibration

Calibration asks whether autonomy follows measured error. We cut each run into windows of 100 actions. Within each class, we correlate the share of actions the controller admitted autonomously with the agent's observed violation rate in the same window. We use the admission decision rather than the executed outcome, so that blocked violations cannot lower the autonomy measure by themselves. Pooling the reliable, drifting and injection-prone agents, EBD shows a clear negative relationship: Spearman $\rho = -0.55$ for class 1 and -0.30 for class 2. Class 3 is near zero (-0.02) because it rarely earns autonomy in the first place. Fixed-budget EBD ($+0.01$, $+0.00$) and static-narrow (-0.00, $+0.00$) show no relationship, as expected. For static-narrow, class 3 is undefined because its autonomy never changes (Figure 12). So P5 holds wherever autonomy can be earned.

7.9. Risk misestimation and reviewer fatigue

Risk misestimation. Figure 13(a) varies the bias b in the controller's risk estimate ($\hat{r} = r/b$, so $b > 1$ means risk is underestimated).

Loss rises steadily but does not collapse. Fleet loss goes from 112 at $b = 1$ to 162 at $b = 8$, while escalation falls from 49% to 25%. The sleeper is hit harder (95 to 223). Overestimating risk ($b < 1$) costs little in loss (106 at $b = 0.25$) but costs oversight (62% escalation).

This is the caveat to P1: the guarantee is only as good as κ in A2, although the verification layers soften moderate errors.

Reviewer fatigue. We model the reviewer's catch probability as $h(1 - \phi\bar{e})$, where $\bar{e}$ is a moving average of recent review load. Figure 13(b) shows that the benefit of reviewing everything is fragile. With no fatigue ($\phi = 0$), always-approve and EBD are equivalent (121 and 120). At $\phi = 0.5$, always-approve loses 1,235, against 523 for EBD and 643 for static-narrow. EBD does better because it asks reviewers to do less. At high fatigue ($\phi \geq 0.75$; at $\phi = 0.9$ it is 826 against 1,018), static-narrow overtakes EBD: its lighter load keeps reviewers sharper, while EBD still escalates about half of all actions. EBD reduces the dependence on reviewer quality but does not remove it.

Jack Sparrow Publishers © 2026, IJRDES, All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6, 2026 , https://doi.org/10.63328/IJRDES-V8RI1P4

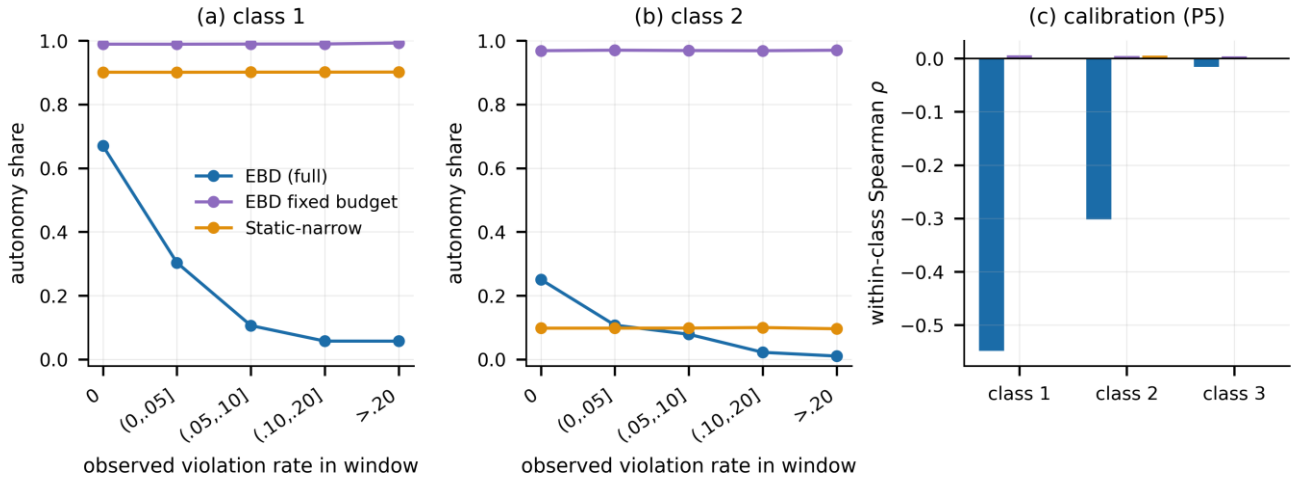

Figure. 12 Calibration. (a, b) Share of actions admitted autonomously against observed violation rate in 100-action windows, for classes 1 and 2. (c) Within-class Spearman correlation. Only the adaptive budget lowers autonomy when error rises.

Table. 7 Tolerable-violation-rate analysis for the reliable agent ($\alpha = 0.03$, $\beta = 1.5$, $\bar{w} = 0.88$). p_{eff} is the violation probability including injected content, δ comes from Lemma 1, and $\nu = p_{\text{eff}}\delta$. "Margin" is $\alpha\bar{w} - \beta\nu$, the expected log-budget drift per autonomous action ignoring decay.

Class	p_{eff}	δ	ν	ν^*	Margin
0	0.30%	0.65	0.19%	1.76%	+0.0235
1	0.49%	0.72	0.36%	1.76%	+0.0211
2	0.88%	0.83	0.73%	1.76%	+0.0154
3	1.57%	0.90	1.42%	1.76%	+0.0052

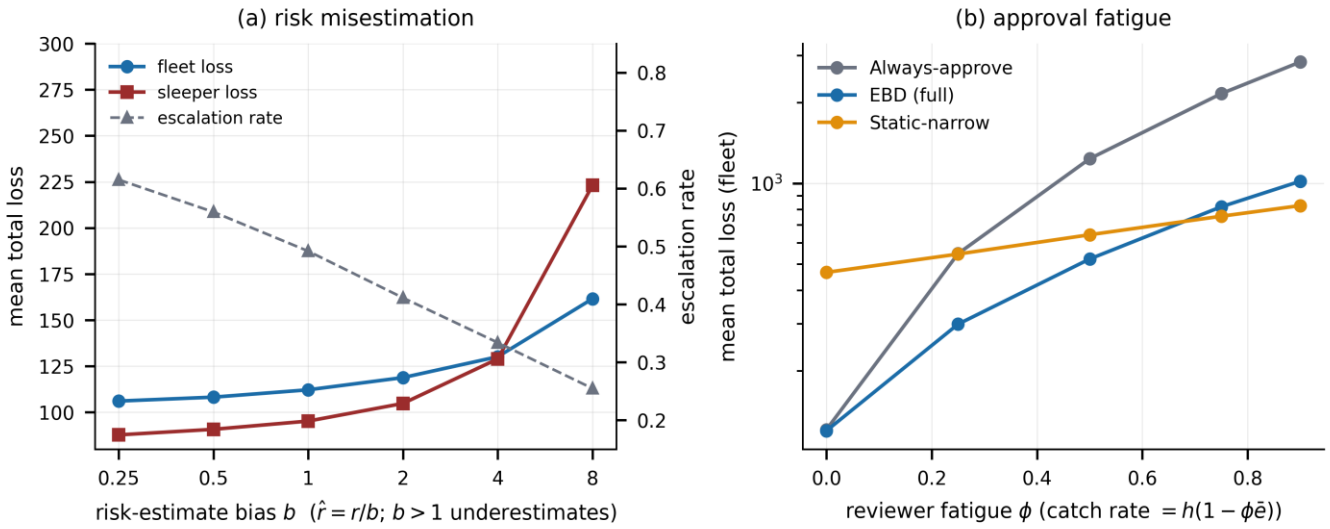

Figure. 13 Robustness. (a) Risk-estimate bias: underestimating risk raises loss and lowers escalation. (b) Reviewer fatigue: always-approve degrades fastest, and at high fatigue the lighter review load of static-narrow overtakes EBD.

7.10. Detection delay and time to demotion

Figure 14 shows how fast EBD takes autonomy away from a sleeper. We measure the time for the class-2 budget to halve after activation. The median is 21 steps for full EBD (90th percentile 71), against 28 without outcome verification and 37 without the gateway. With the symmetric response ($\beta = \alpha$), 100% of runs are never demoted within the horizon. The simulated outcome-detection delay follows the truncated exponential used to generate it (Figure 14b), which is a useful sanity check on the delay queue.

7.11. Overhead and a simple cost model

Latency is dominated by human review. With a modelled 30 s reviewer wait, EBD adds 14.7 s per action on average, against 30.0 s for always-approve (Figure 15a). The

Jack Sparrow Publishers © 2026, IJRDES, All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6, 2026 , https://doi.org/10.63328/IJRDES-V8RI1P4

gateway's own share is milliseconds. Outcome verification samples about 38% of actions, and that compute is the price of L4. To put the policies on one scale, we define an operating cost , where is the number of reviews per run and is the cost of one review in loss units. Table 8 evaluates for several values of . At low review cost ( ) EBD is cheapest. EBD beats static-narrow while and static-broad while . It beats the fixed-budget and symmetric variants only while and respectively. When reviews are expensive, simpler policies win. Where to operate is therefore a business and risk decision, and the framework's contribution is to make that decision explicit.

Table 8: Operating cost $J = \text{loss} + c_H \times \text{reviews}$ for review cost $c_H \in \{0.05, 0.25, 0.5, 1\}$ loss units (fleet means; bold marks the lowest in each column).

Policy	Reviews/run	$c_H = 0.05$	0.25	0.5	1
Always-approve	2,000	214	614	1,114	2,114
Static-narrow	525	486	591	722	985
Static-broad	124	1,567	1,592	1,623	1,685
Post-hoc audit	0	3,429	3,429	3,429	3,429
EBD (full)	978	**162**	**358**	602	1,091
EBD fixed budget	161	537	569	609	**690**
EBD $\beta = \alpha$	318	459	523	**602**	761

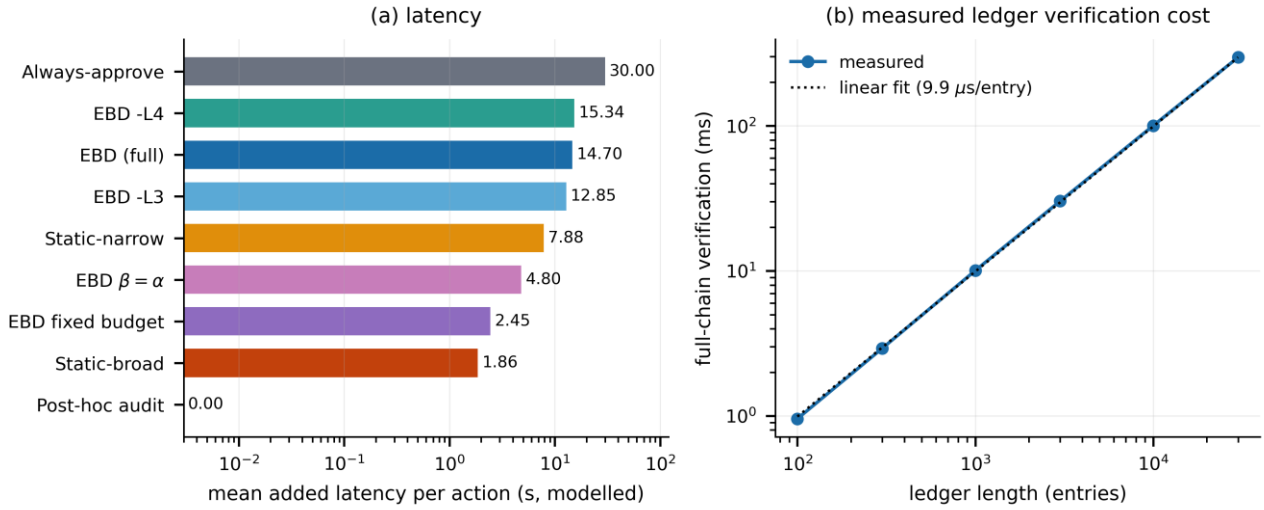

Figure. 15 (a) Modelled mean added latency per action. Human review dominates.
(b) Measured full-chain ledger verification time from the prototype, which grows linearly with chain length

.

8. Prototype and Measured Evidence

To ground the simulation we implemented the gateway, a hash-chained and authenticated ledger, and the budget controller in Python (roughly 100 lines for the core classes). Each ledger record includes the hash of the previous record, is hashed with SHA-256 and is authenticated with HMAC-SHA256. Verification recomputes every hash, checks linkage and indices, and validates every authentication tag. It can also check an externally held signed checkpoint (a length and a head hash). We took the measurements on the sandbox machine used for this work, on one thread and without tuning, so treat them as order-of-magnitude figures, not benchmarks.

Speed. Over 100,000 synthetic actions, a full gateway decision plus signed ledger append took a median of 10.4 μs (95th percentile 19.7 μs, 99th 38.6 μs). That is a sustained rate of about 78,459 actions per second on one thread. The decision alone takes about 0.28 μs (Figure 16a). Verifying the whole chain grows linearly at about 9.9 μs per entry, or 297 ms for 30,000 entries (Figure 15b). These costs are small next to human review or model inference.

Tamper evidence (P3). We ran 1,000 trials of each attack on a 500-record chain (Figure 16b). Every in-place change was detected in all trials. That includes recomputing the whole hash chain without the key, deleting a record, swapping two records and inserting a forged one. Cutting records off the end of the log, however, went undetected in every trial when no external anchor was kept. With a signed checkpoint held by an independent party, detection

Jack Sparrow Publishers © 2026, IJRDES, All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6, 2026 , https://doi.org/10.63328/IJRDES-V8RI1P4

was complete. This is why Section 4 calls for external anchoring: a hash chain alone shows tampering with what remains but cannot prove nothing was removed from the end.

What the prototype shows. It supports P2, since each record carries the agent identifier and is authenticated, and P3 for the attacks listed. It also shows that gateway mediation is cheap. It does not test a real model, real tools or key management. HMAC uses a shared key, so anyone holding the key can forge records. A real deployment would use asymmetric signatures with hardware-protected keys.

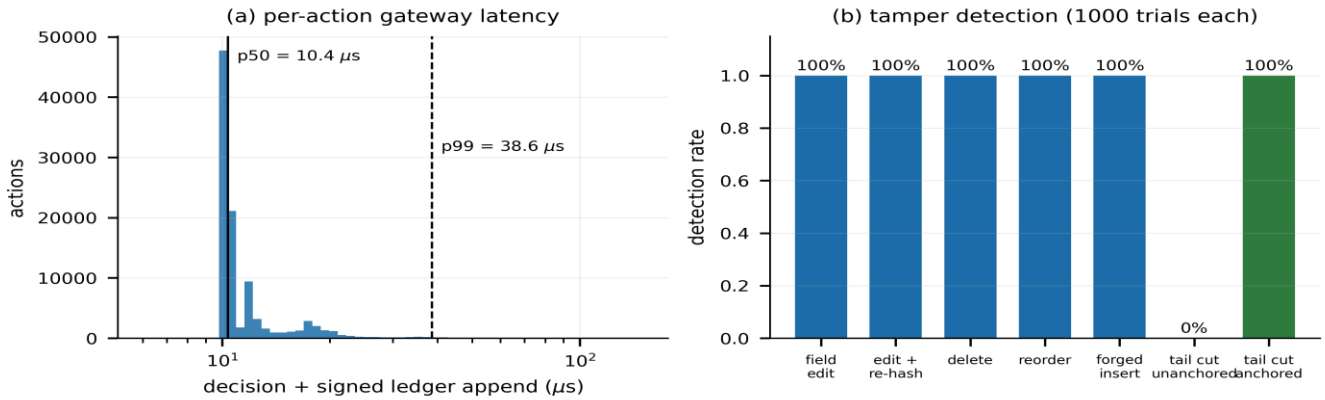

Figure. 16 Prototype measurements. (a) Distribution of per-action latency for decision plus signed ledger append (100,000 actions). (b) Tamper-detection rates. Unanchored tail truncation is undetectable by construction

Discussion and Limitations

The biggest limit is that every performance number comes from a simulator. Its agents, detectors and reviewers are simplified, and its parameters are assumptions (Appendix A). The absolute values, such as loss units and detection probabilities, should not be carried over to real systems. What should carry over are the mechanisms and trade-offs: asymmetric response contains adaptive adversaries and fixed grants do not, the savings in oversight depend on how reviewers behave, and a cap on high-risk classes keeps them under human control. Checking these claims on real agents with real incident data is the most important piece of future work. Some limits come from the assumptions behind the theory. Verification is only as good as the specification, and a contract cannot capture all intent. In our model that shows up as the share $1 - f$ of violations the gateway cannot see. Lemma 1 and Theorem 1 assume independent detection with a known lower bound. If detectors share blind spots, for example a reviewer model from the same family as the agent, the product form no longer holds, and a compromised verifier breaks A3 outright. The bound itself held in every setting we tried but is loose by two to three orders of magnitude, so it should guide the choice of penalty ratio and not be used to predict loss.

Two weaknesses are built into the design. Because undetected violations look clean, a patient adversary can build up budget with violations that evade every detector. Slow rewards, fast penalties, per-class budgets, hard caps and diverse evidence limit this but cannot remove it, and in our experiments the cap on class 3 is the last line of defence. The second is human. Our fatigue model is simple. EBD lowers the review load, but at high fatigue a lighter static policy can do better, and real reviewers vary in ways we did not model, such as anchoring, rubber-stamping and being shown actions in misleading ways.

Finally, we analysed one agent per class. Delegation from one agent to another, and collusion between agents, are open. Logging every action also raises privacy concerns, so ledger contents should be kept to a minimum and access controlled. Someone has to be named as responsible when a verified agent still causes harm.

9. Conclusion

We proposed Evidence-Based Delegation, which ties an agent's authority to layered, verifiable evidence through a log-scale trust budget that rewards slowly and penalises sharply. The theory bounds what an adaptive adversary can take and shows how promotion time and tolerable error depend on the penalty ratio. In simulation, EBD matched the loss of reviewing every action with about half the reviews, did far better than static, fixed-budget, symmetric and audit-only alternatives against drifting and sleeper agents, and gave autonomy that followed measured error. The same experiments showed where it falls short. It escalates heavily for unreliable agents, it is sensitive to underestimated risk, it still depends on the quality of reviewers, and its formal bound is loose. The prototype showed that gateway mediation costs microseconds, and that a log needs an external anchor if it is to prove nothing was cut from the end. The next steps are to test the framework with real

Jack Sparrow Publishers © 2026, IJRDES, All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6, 2026 , https://doi.org/10.63328/IJRDES-V8RI1P4

language-model agents and incident data, to learn risk estimators, to treat delayed detection and correlated verifiers formally, to study trust across chains of agents, and to test approval interfaces with human participants.

References

[1]. D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, D. Mané. Concrete problems in AI safety. arXiv:1606.06565, 2016.

[2]. M. Bellare, R. Canetti, H. Krawczyk. Keying hash functions for message authentication. In *CRYPTO*, 1996.

[3]. S. Goldwasser, S. Micali, C. Rackoff. The knowledge complexity of interactive proof-systems. In *Proc. 17th ACM STOC*, 1985.

[4]. K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, M. Fritz. Not what you've signed up for: compromising real-world LLM-integrated applications with indirect prompt injection. arXiv:2302.12173, 2023.

[5]. D. Hadfield-Menell, A. Dragan, P. Abbeel, S. Russell. The off-switch game. In *IJCAI*, 2017.

[6]. L. Lamport, R. Shostak, M. Pease. The Byzantine generals problem. *ACM TOPLAS*, 4(3):382–401, 1982.

[7]. J. D. Lee, K. A. See. Trust in automation: designing for appropriate reliance. *Human Factors*, 46(1):50–80, 2004.

[8]. M. Leucker, C. Schallhart. A brief account of runtime verification. *J. Logic and Algebraic Programming*, 78(5):293–303, 2009.

[9]. R. C. Mayer, J. H. Davis, F. D. Schoorman. An integrative model of organizational trust. *Academy of Management Review*, 20(3):709–734, 1995.

[10]. L. Ouyang et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

[11]. R. Parasuraman, V. Riley. Humans and automation: use, misuse, disuse, abuse. *Human Factors*, 39(2):230–253, 1997.

[12]. T. Schick et al. Toolformer: language models can teach themselves to use tools. In *NeurIPS*, 2023.

[13]. F. B. Schneider. Enforceable security policies. *ACM TISSEC*, 3(1):30–50, 2000.

[14]. F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[15]. S. Yao et al. ReAct: synergizing reasoning and acting in language models. In *ICLR*, 2023.

How To Cite:

Marpu Jagadeesh , " Trustworthy Autonomous Agents Through Evidence-Based Delegation: Formal Verification, Prototype Implementation, and Simulation Analysis", International Journal of Research and Development in Engineering Sciences(IJRDES), Volume 8 , Issue 1 , pp 31 – 48 , November 2026.

CrossRef DOI: https://doi.org/10.63328/ijrdes-v8ri6p4

Appendix A. Simulation parameters

Table. 9 All simulation parameters. These are modelling assumptions, not measurements of real systems.

Parameter	Value
Horizon, seeds	$T = 2{,}000$ actions; $1{,}000$ seeds (main), 200–500 (sweeps); common random numbers
Classes	$K = 4$; $\pi = (0.45, 0.30, 0.18, 0.07)$; median risk $(1, 5, 20, 100)$; log-sd 0.5
Base violation prob.	$(0.002, 0.004, 0.008, 0.015)$
Mitigable fraction μ_c	$(0.9, 0.7, 0.4, 0.1)$, decays with detection delay (time constant 50)
L4 sampling s_c	$(0.2, 0.4, 0.7, 0.9)$
Detection	Gateway $d_g = 0.97$ on contract-detectable share $f = 0.6$; outcome $d_o = 0.85$; reviewer $h = 0.9$, false rejection 0.02; gateway false-positive escalation 0.01
Detection delay	Exponential, mean 15 actions, truncated at 200
Evidence strengths	$w_g = 0.15$, $w_o = 0.85$, $w_h = 0.90$
Controller (full EBD)	$\alpha = 0.03$, $\beta = 1.5$, $\gamma = 3 \times 10^{-4}$, $B_0 = 2$, $B^{\min} = 0.5$; caps $(30, 60, 120, 80)$; risk-estimate noise log-sd 0.2, bias $b = 1$
Static baselines	Thresholds 10 and 60 risk units
Audit baseline	10% audit rate, detection 0.85
Fixed-budget ablation	Budgets $(15, 30, 60, 40)$
Agents	Drift onset 600, ramp to $12\times$; sleeper onset 500, attack prob. 0.5; injection rates 2% / 8%, susceptibility 0.05 / 0.5
Latency, fatigue	Reviewer wait 30 s; gateway 40 ms; catch rate $h(1 - \phi\bar{e})$, $\bar{e}$ = recent review load (window ≈ 200)

Jack Sparrow Publishers © 2026, IJRDES, All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6, 2026 , https://doi.org/10.63328/IJRDES-V8RI1P4

Appendix B. Reproducibility and figure provenance

All results come from the accompanying code (simulator, experiment driver, prototype, figure and table generators) with fixed seeds. Every figure, including the flowcharts, was generated from code using simulation or prototype output, and none was copied from another source. Figures are stored at 300 dpi (1950 pixels at 6.5 in) and printed at or below that width, so the effective resolution is at least 300 dpi. Numbers in the text and tables are filled in by script from the result files.

Declaration

Conflicts of Interest: The authors declare no conflict of interest.

Author Contribution: All authors wrote the main manuscript text and also consent to the submission.

Ethical approval: Not applicable.

Consent to Participate: All authors consent to participate.

Funding: Not applicable, and No funding was received

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Personal Statement: We declare with our best of knowledge that this research work is purely Original Work and No third party material used in this article drafting. If any such kind material found in further online publication, we are responsible only for any judicial and copyright issues.

Generative AI Statement: The authors declare that generative AI was not used in the preparation or creation of this manuscript. The alternative text (alt text) accompanying the figures in this article was generated by Frontiers with the assistance of artificial intelligence. Reasonable efforts have been made to ensure the accuracy and appropriateness of the generated alt text, including review by the authors wherever possible. If you identify any inaccuracies or issues, please contact the editorial team.

Publisher's Note: The statements, opinions, and claims presented in this article are exclusively those of the authors and do not necessarily reflect the views of their affiliated institutions, the publisher, editors, or reviewers. Any products discussed or evaluated, as well as any claims made by their manufacturers, are not warranted, approved, or endorsed by the publisher.

Acknowledgements

We thank everyone who inspired our work

Jack Sparrow Publishers © 2026, IJRDES, All Rights Reserved
www.jacksparrowpublishers.com